The Am9513 System Timing Controller

Handbook

Advanced Micro Devices

Am9513
System Timing Controller
Technical Manual

The International Standard of Quality guarantees these electrical AQLs on all parameters over the operating temperature range: 0.1% on MOS RAMs & ROMs; 0.2% on Bipolar Logic & Interface; 0.3% on Linear, LSI & Logic & other memories.

INT'STD-123

Printed in U.S.A. 5/83 03402B-MMP

TABLE OF CONTENTS

	Page
PREFACE	i
CHAPTER 1 – THE Am9513	
Introduction	1-1
Functional Description	1-2
Interface Signal Description	1-3
Control Port Registers	1-5
Command Register	1-5
Data Pointer Register	1-5
Prefetch Circuit	1-7
Status Register	1-8
Data Port Registers	1-8
Counter Logic Groups	1-8
Load Register	1-8
Hold Register	1-8
Counter Mode Register	1-8
Alarm Registers and Comparators	1-8
Master Mode Control Options	1-8
Counter Mode Operating Descriptions	1-11
Counter Mode Control Options	1-22
Command Descriptions	1-25
CHAPTER 2 – Am9513 INTERFACING	
Am9513 – CPU Interfacing	2-1
Clock Generation	2-1
Register Access	2-3
Information Transfer Protocols	2-3
Software Initialization	2-3
Command Initiation	2-3
Setting the Data Pointer Register	2-4
Reading the Status Register	2-6
Reading from the Data Port	2-7
Writing to the Data Port	2-8
CHAPTER 3 – CONCATENATING COUNTERS	3-1
CHAPTER 4 – TIME-OF- DAY COUNTING	
Initializing to Current Time-of-Day	4-1
Reading the Current Time	4-3
Setting the Alarm Time	4-3
Other Time-of-Day Variations	4-3
Am8080A/8085A Time-of-Day Software	4-4
A Cookbook Approach to Time-of-Day Counting	4-7
Settime Software Using Macros	4-9
Console Driven Clock Runs under CP/M	4-9
CHAPTER 5 – EVENT COUNTING	5-1
CHAPTER 6 – FREQUENCY AND BAUD RATE GENERATION	
Frequency Generation	6-1
Auto Baud Rate Generator	6-3
CHAPTER 7 – ONE-SHOT APPLICATIONS	7-1
CHAPTER 8 – SOFTWARE CONSIDERATIONS AND PROGRAM EXAMPLES	8-1
APPENDICES	
A – Dealing with Metastable Problems	A-1
B – Key to Timing Diagrams	B-1
C – Am9513 Software Definitions	C-1

TABLE OF CONTENTS (Cont.)

D – Am9513 Macro Command Summary D-1
E – Am9513 Macros for Am8080/Am8085 ..... E-1
F – Am9513 Macros for Z80 ..... F-1
G – Am9513 Macros for Z8000 ..... G-1
H – Am9513 C Definitions H-1
I – Am9513 Assembler Definitions I-1

PREFACE

This manual describes the functional operation of the Am9513 System Timing Controller and its typical hardware and software applications.

Due to the complexity of this device, the first two chapters of this book are required reading before attempting to use the device. Detailed timing information is not contained in this manual, it is published in a separate document, called "Am9513 Electrical Specification," and is available from any Advanced Micro Devices Sales Office, representative or franchised distributor or directly from AMD Literature Distribution (MS-82) P.O. Box 453, Sunnyvale, CA 94086.

The AmZ8073 is functionally identical to the Am9513, but offers timing parameters that are optimized for operation with the AmZ8000* microprocessor.

The Am9513A is a functionally enhanced version of the Am9513, and fully compatible with the Am9513. The new, additional features of this device are pointed out in the text.

*Z8000 is a trademark of Zilog, Inc.

Chapter 1
The Am9513

Chapter 1
The Ams513

INTRODUCTION

Manipulation and coordination of timing parameters and event sequences are universal system attributes. At the most fundamental levels of control, time sequences are intimately embedded in the essential hardware and interface concepts of all processors: the necessary flows of step-by-step procedures are inherent in the execution of even the most basic programs. At the interface level, both internal and external hardware coordination usually require several types of timing-oriented exchanges. In general, control of system and sub-system processes will often involve sophisticated levels of counting, sequencing and timing manipulations. The specific mix of such activities will, of course, be application dependent, yet counting/timing concepts are at least fundamentally involved in all system operations, from the simplest sequencing of a hardware interface to the complex interaction of high-level processes.

Time-related activities fall into a wide variety of categories. Frequency generation, waveform duty cycle control, event counting, interval measurement, precise periodic interrupts, time-of-day accumulation, delays, gap detection, etc., are just a few of the types of operations typically undertaken. When the system must accomplish several of these activities, especially when some measure of concurrency is necessary, a significant portion of the available processing and/or hardware logic resources can be consumed. Throughput limitations easily arise.

A specialized circuit with enough versatility to handle many types of counting and timing functions would therefore be able to simplify software, improve system performance and decrease system chip count. The Am9513 System Timing Controller has been designed to accomplish just such a task. It provides significant capability for waveform generation, counting, timing and intervalometer functions for many types of processor-oriented systems. It offers an unusually versatile control structure that allows the use of many operating configurations so that a wide variety of applications can be efficiently serviced.

The operating philosophy of the Am9513 is based on the use of general-purpose counters that can be controlled in various ways to produce the functions desired. Broadly, use of the counters falls into two classic categories: (a) count accumulation, and (b) frequency division.

In the first case, the counter simply accumulates a count of transitions that occur on its input. An output that indicates the zero state of the counter would be of only incidental interest. The counter value should be available at any time to the associated CPU or it might be compared with some independent value. The accumulated count might be modified or the counter input conditioned by various controls, including hardware and software gating functions; in any event, in these types of applications, it is the value of the actual count that is of interest.

In the case of frequency division, on the other hand, it is an output waveform that is of interest and the counter input information may be incidental. With an output signal that indicates the zero state of the counter, selection of the effective length of the counter and the input frequency are controlled to provide the desired output frequency. Additional controls may allow various types of output waveforms to be generated from the base output frequency, but the actual counter value will usually not be of direct interest.

The Am9513 has been designed to handle effectively both modes of operation, even intermixed on the same chip. In many instances, of course, both types of counter usage will be combined to provide the desired function.

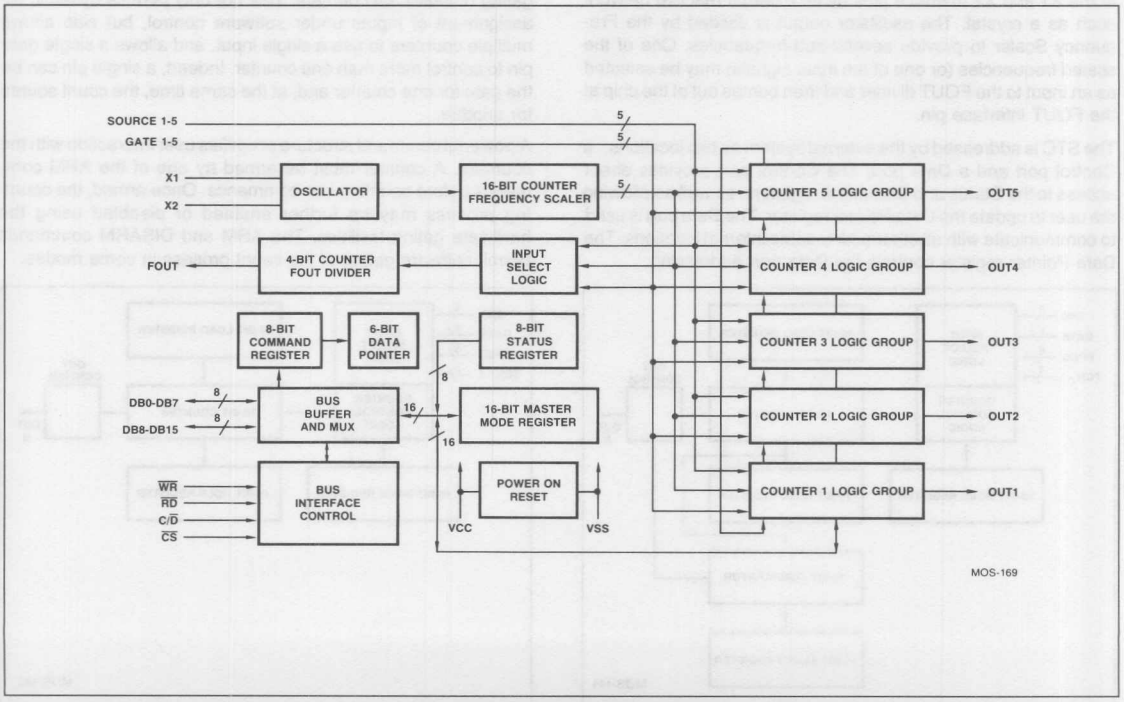


Figure 1-1. General Block Diagram

FUNCTIONAL DESCRIPTION

The Am9513 System Timing Controller (STC) is a support device for processor oriented systems that is designed to enhance the available capability with respect to counting and timing operations. It provides the capability for programmable frequency synthesis, high resolution programmable duty cycle waveforms, retriggerable digital timing functions, time-of-day clocking, coincidence alarms, complex pulse generation, high resolution baud rate generation, frequency shift keying, stop-watching timing, event count accumulation, waveform analysis and many more. A variety of programmable operating modes and control features allow the Am9513 to be personalized for particular applications as well as dynamically reconfigured under program control.

The STC includes five general-purpose 16-bit counters. A variety of internal frequency sources and external pins may be selected as inputs for individual counters with software selectable active-high or active-low input polarity. Both hardware and software gating of each counter is available. Three-state outputs for each counter provide either pulses or levels. The counters can be programmed to count up or down in either binary or BCD. The accumulated count may be read without disturbing the counting process. Any of the counters may be internally concatenated to form an effective counter length of up to 80 bits.

The Am9513 block diagrams (Figures 1-1, 1-2 and 1-3) indicate the interface signals and the basic flow of information. Internal control lines and the internal data bus have been omitted. The control and data registers are all connected to a common internal 16-bit bus. The external bus may be 8- or 16-bits wide; in the 8-bit mode the internal 16-bit information is multiplexed to the low order data bus pins DB0 through DB7.

An internal oscillator provides a convenient source of frequencies for use as counter inputs. The oscillator's frequency is controlled at the X1 and X2 interface pins by an external reactive network such as a crystal. The oscillator output is divided by the Frequency Scaler to provide several sub-frequencies. One of the scaled frequencies (or one of ten input signals) may be selected as an input to the FOUT divider and then comes out of the chip at the FOUT interface pin.

The STC is addressed by the external system as two locations: a Control port and a Data port. The Control port provides direct access to the Status and Command registers, as well as allowing the user to update the Data Pointer register. The Data port is used to communicate with all other addressable internal locations. The Data Pointer register controls the Data port addressing.

Among the registers accessible through the Data port are the Master Mode register and five Counter Mode registers, one for each counter. The Master Mode register controls the programmable options that are not controlled by the Counter Mode registers.

Each of the five general-purpose counters is 16-bits long and is independently controlled by its Counter Mode register. Through this register, a user can software select one of 16 sources as the counter input, a variety of gating and repetition modes, up or down counting in binary or BCD and active-high or active-low input and output polarities.

Associated with each counter are a Load register and a Hold register, both accessible through the Data port. The Load register is used to automatically reload the counter to any predefined value, thus controlling the effective count period. The Hold register is used to save count values without disturbing the count process, permitting the host processor to read intermediate counts. In addition, the Hold register may be used as a second Load register to generate a number of complex output waveforms.

All five counters have the same basic control logic and control registers. Counters 1 and 2 have additional Alarm registers and comparators associated with them, plus the extra logic necessary for operating in a 24-hour time-of-day mode. For real-time operation the time-of-day logic will accept 50Hz, 60Hz or 100Hz input frequencies.

Each general counter has a single dedicated output pin. It may be turned off when the output is not of interest or may be configured in a variety of ways to drive interrupt controllers, Darlington buffers, bus drivers, etc. The counter inputs, on the other hand, are specifically not dedicated to any given interface line. Considerable versatility is available for configuring both the input and the gating of individual counters. This not only permits dynamic reassignment of inputs under software control, but also allows multiple counters to use a single input, and allows a single gate pin to control more than one counter. Indeed, a single pin can be the gate for one counter and, at the same time, the count source for another.

A powerful command structure simplifies user interaction with the counters. A counter must be armed by one of the ARM commands before counting can commence. Once armed, the counting process may be further enabled or disabled using the hardware gating facilities. The ARM and DISARM commands permit software gating of the count process in some modes.

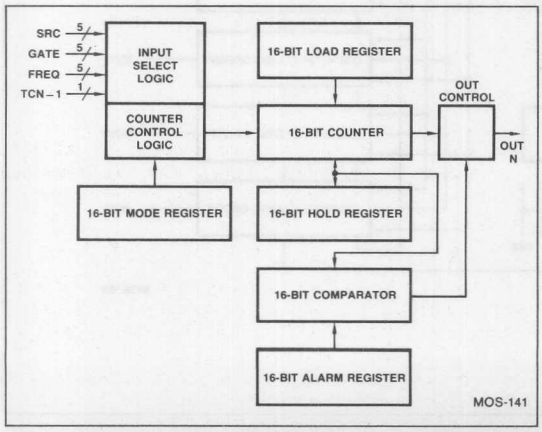


Figure 1-2. Counter Logic Groups 1 and 2

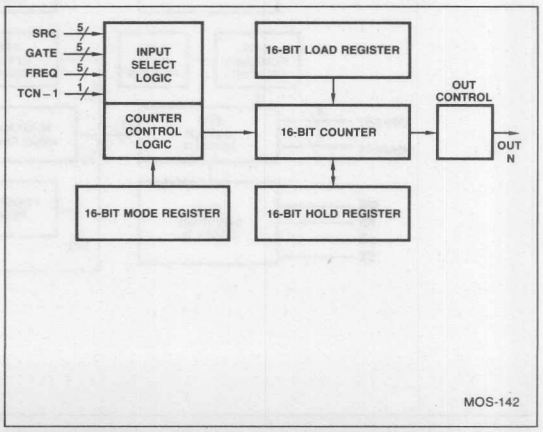


Figure 1-3. Counter Logic Groups 3, 4 and 5

The LOAD command causes the counter to be reloaded with the value in either the associated Load register or the associated Hold register. It will often be used as a software retrigger or as counter initialization prior to active hardware gating.

The DISARM command disables further counting independent of any hardware gating. A disarmed counter may be reloaded using the LOAD command, may be incremented or decremented using the STEP command and may be read using the SAVE command. A count process may be resumed using an ARM command.

The SAVE command transfers the contents of a counter to its associated Hold register. This command will overwrite any previous Hold register contents. The SAVE command is designed to allow an accumulated count to be preserved so that it can be read by the host CPU at some later time.

Two combinations of the basic commands exist to either LOAD AND ARM or to DISARM AND SAVE any combination of counters. Additional commands are provided to: step an individual counter by one count; set and clear an output toggle; issue a software reset; clear and set special bits in the Master Mode register; and load the Data Pointer register.

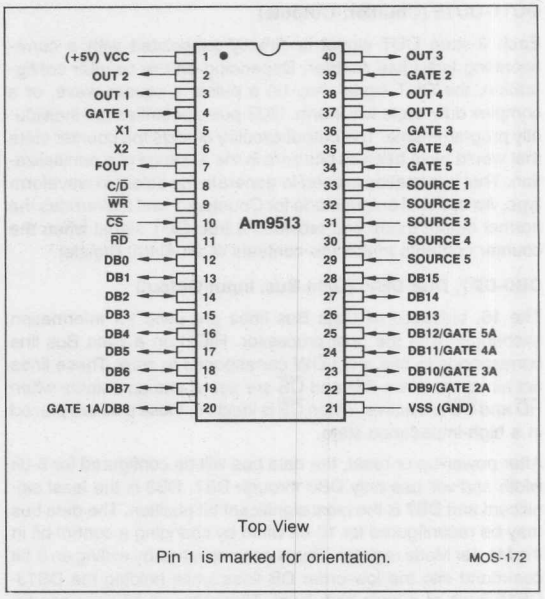


Figure 1-4. Connection Diagram

INTERFACE SIGNAL DESCRIPTION

Figure 1-5 summarizes the interface signals and their abbreviations for the STC. Figure 1-4 shows the signal pin assignments for the standard 40-pin dual in-line package.

VCC: +5 volt power supply

VSS: Ground

X1, X2 (Crystal)

X1 and X2 are the connections for an external crystal used to determine the frequency of the internal oscillator. The crystal should be a parallel-resonant, fundamental-mode type. An RC or LC or other reactive network may be used instead of a crystal. For driving from an external frequency source, X1 should be left open and X2 should be connected to a TTL source and a pull-up resistor.

FOUT (Frequency Out, Output)

The FOUT output is derived from a 4-bit counter that may be programmed to divide its input by any integer value from 1 through 16 inclusive. The input to the counter is selected from any of 15 sources, including the internal scaled oscillator frequencies. FOUT may be gated on and off under software control and when off will exhibit a low impedance to ground. Control over the various FOUT options resides in the Master Mode register. After power-up, FOUT provides a frequency that is 1/16 that of the oscillator.

GATE1-GATE5 (Gate, Inputs)

The Gate inputs may be used to control the operations of individual counters by determining when counting may proceed. The same Gate input may control up to three counters. Gate pins may also be selected as count sources for any of the counters and for the FOUT divider. The active polarity for a selected Gate input is programmed at each counter. Gating function options allow level-sensitive gating or edge-initiated gating. Other gating modes are available including one that allows the Gate input to select between two counter output frequencies. All gating functions may also be disabled. The active Gate input is conditioned by an auxiliary input when the unit is operating with an external 8-bit data bus. See Data Bus description. Schmitt-trigger circuitry on the GATE inputs allows slow transition times to be used.

Signal	Abbreviation	Type	Pins
+5 Volts	VCC	Power	1
Ground	VSS	Power	1
Crystal	X1, X2	I/O, I	2
Read	$\overline{RD}$	Input	1
Write	$\overline{WR}$	Input	1
Chip Select	$\overline{CS}$	Input	1
Control/Data	C/$\overline{D}$	Input	1
Source N	SRC	Input	5
Gate N	GATE	Input	5
Data Bus	DB	I/O	16
Frequency Out	FOUT	Output	1
Out N	OUT	Output	5

Figure 1-5. Interface Signal Summary

SRC1-SRC5 (Source, Inputs)

The Source inputs provide external signals that may be counted by any of the counters. Any Source line may be routed to any or all of the counters and the FOUT divider. The active polarity for a selected SRC input is programmed at each counter. Any duty cycle waveform will be accepted as long as the minimum pulse width is at least half the period of the maximum specified counting frequency for the part. Schmitt-trigger circuitry on the SRC inputs allows slow transition times to be used.

OUT1-OUT5 (Counter, Outputs)

Each 3-state OUT signal is directly associated with a corresponding individual counter. Depending on the counter configuration, the OUT signal may be a pulse, a square wave, or a complex duty cycle waveform. OUT pulse polarities are individually programmable. The output circuitry detects the counter state that would have been all bits zero in the absence of a reinitialization. That information is used to generate the selected waveform type. An optional output mode for Counters 1 and 2 overrides the normal output mode and provides a true OUT signal when the counter contents match the contents of an Alarm register.

DB0-DB7, DB8-DB15 (Data Bus, Input/Output)

The 16, bidirectional Data Bus lines are used for information exchanges with the host processor. HIGH on a Data Bus line corresponds to one and LOW corresponds to zero. These lines act as inputs when $\overline{WR}$ and $\overline{CS}$ are active and as outputs when $\overline{RD}$ and $\overline{CS}$ are active. When $\overline{CS}$ is inactive, these pins are placed in a high-impedance state.

After power-up or reset, the data bus will be configured for 8-bit width and will use only DB0 through DB7. DB0 is the least significant and DB7 is the most significant bit position. The data bus may be reconfigured for 16-bit width by changing a control bit in the Master Mode register. This is accomplished by writing an 8-bit command into the low-order DB lines while holding the DB13-DB15 lines at a logic high level. Thereafter all 16 lines can be used, with DB0 as the least significant and DB15 as the most significant bit position.

When operating in the 8-bit data bus environment, DB8-DB15 will never be driven active by the Am9513. DB8 through DB12 may optionally be used as additional Gate inputs (see Figure 1-6). If unused they should be held high. When pulled low, a GATENA signal will disable the action of the corresponding counter N gating. DB13-DB15 should be held high in 8-bit bus mode whenever $\overline{CS}$ and $\overline{WR}$ are simultaneously active.

$\overline{CS}$ (Chip Select, Input)

The active-low Chip Select input enables Read and Write operations on the data bus. When Chip Select is high, the Read and Write inputs are ignored. The first Chip Select signal after power-up is used to clear the power-on reset circuitry. If Chip Select is tied to ground permanently, the power-on reset circuitry may not function. In such a configuration, the software reset command must be issued following power-up to reset the Am9513.

$\overline{RD}$ (Read, Input)

The active-low Read signal is conditioned by Chip Select and indicates that internal information is to be transferred to the data bus. The source will be determined by the port being addressed and, for Data Port reads, by the contents of the Data Pointer register. $\overline{WR}$ and $\overline{RD}$ should be mutually exclusive.

$\overline{WR}$ (Write, Input)

The active-low Write signal is conditioned by Chip Select and indicates that data bus information is to be transferred to an internal location. The destination will be determined by the port being addressed and, for Data Port writes, by the contents of the Data Pointer register. $\overline{WR}$ and $\overline{RD}$ should be mutually exclusive.

C/$\overline{D}$ (Control/Data, Input)

The Control/Data signal selects source and destination locations for read and write operations on the data bus. Control Write operations load the Command register and the Data Pointer. Control Read operations output the Status register. Data Read and Data Write transfers communicate with all other internal registers. Indirect addressing at the data port is controlled internally by the Data Pointer register.

Package Pin	Data Bus Width (MM14)	
	16 Bits	8 Bits
12	DB0	DB0
13	DB1	DB1
14	DB2	DB2
15	DB3	DB3
16	DB4	DB4
17	DB5	DB5
18	DB6	DB6
19	DB7	DB7
20	DB8	GATE 1A
22	DB9	GATE 2A
23	DB10	GATE 3A
24	DB11	GATE 4A
25	DB12	GATE 5A
26	DB13	(VIH)
27	DB14	(VIH)
28	DB15	(VIH)

Figure 1-6. Data Bus Assignments

Interface Considerations

All of the input and output signals for the Am9513 are specified with logic levels compatible with those of standard TTL circuits. See the Am9513 data sheet for specifications. In addition to providing TTL compatible voltage levels, other output conditions are specified to help configure non-standard interface circuitry. The logic level specifications take into account all worst-case combinations of the three variables that affect the logic level thresholds: ambient temperature, supply voltage and processing parameters. A change in any of these toward nominal values will improve the actual operating margins and will increase noise immunity.

Unprotected open gate inputs of high quality MOS transistors exhibit very high resistances on the order of perhaps 10^{14} ohms. It is easy, therefore, in some circumstances, for charge to enter the gate node of such an input faster than it can be discharged and consequently for the gate voltage to rise high enough to break down the oxides and destroy the transister. All inputs to the Am9513 include protection networks to help prevent damaging accumulations of static charge. The protection circuitry is designed to slow the transistions of incoming current surges and to provide low impedance discharge paths for voltages beyond the normal operating levels. Note, however, that input energy levels can nonetheless be too high to be successfully absorbed. Conventional design, storage, and handling precautions should be observed so that the protection networks themselves are not overstressed.

Within the limits of normal operation, the input protection circuitry is inactive and may be modeled as a lumped series RC as shown in Figure 1-7a. The functionality active input connection during normal operation is the gate of an MOS transistor. No active sources or drains are connected to the inputs so that neither transient nor steady-state currents are impressed on the driving signals other than the charging or discharging of the input capacitance and the accumulated leakage associated with the protection network and the input circuit.

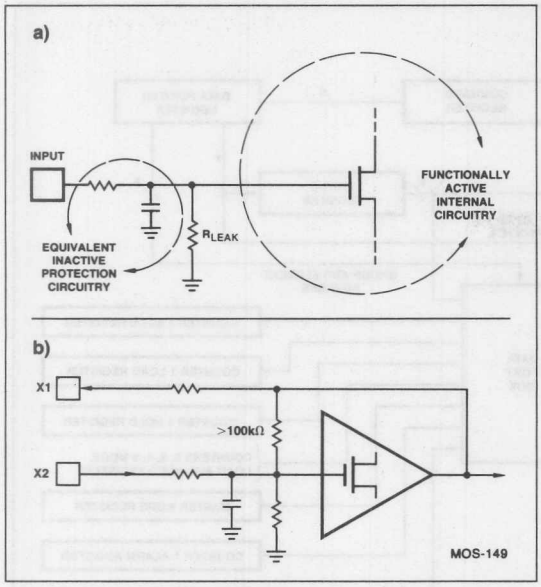


Figure 1-7. Input Circuitry

The only exception to the purely capacitive input case is the X2 crystal input. As shown in Figure 1-7b, an internal resistor connects X1 and X2 in addition to the protection network. The resistor is a modestly high value of more than 100kohms.

Fanout from the driving circuitry into the Am9513 inputs will generally be limited by transition time considerations rather than DC current limitations when the loading is dominated by conventional MOS circuits. In an operating environment, all inputs should be terminated so they do not float and therefore will not accumulate stray static charges. Unused inputs should be tied directly to Ground or VCC, as appropriate. An input in use will have some type of logic output driving it and termination during operation will not be a problem. Where inputs are driven from logic external to the card containing this chip, however, on-board termination should be provided to protect the chip when the board is unplugged and the input would therefore otherwise float. A pull-up resistor or a simple inverter or gate will suffice.

Power Supply

The Am9513 requires only a single 5V power supply. Maximum supply currents are specified in the electrical specification at the high end of the voltage tolerance and the low end of the temperature range. In addition, the current specifications take into account the worstcase distribution of processing parameters that may be encountered during the manufacturing life of the product. Typical supply current values, on the other hand, are specified at a nominal +5.0 volts, a nominal ambient temperature of 25°C, and nominal processing parameters. Supply current always decreases with increasing ambient temperature: thermal run-away is not a problem.

Supply current will vary somewhat from part to part, but a given unit at a given operating temperature will exhibit a nearly constant power drain. There is no functional operating region that will cause more than a few percent change in the supply current. Decoupling of VCC, then, is straightforward and will generally be used to isolate the Am9513 from VCC noise originating externally.

CONTROL PORT REGISTERS

The STC is addressed by the external system as only two locations: a Control port and a Data port. Transfers at the Control port ($C/\overline{D}$ = High) allow direct access to the command register when writing and the status register when reading. All other available internal locations are accessed for both reading and writing via the Data port ($C/\overline{D}$ = Low). Data port transfers are executed to and from the location currently addressed by the Data Pointer register. Options available in the Master Mode register and the Data Pointer control structure allow several types of transfer sequencing to be used. See Figure 1-8.

Transfers to and from the Control port are always 8-bits wide. Each access to the Control port will transfer data between the Command register (writes) or Status register (reads) and Data Bus pins DB0-DB7, regardless of whether the Am9513 is in 8- or 16-bit bus mode. When the Am9513 is in 8-bit bus mode, Data Bus pins DB13-DB15 should be held at a logic high whenever $\overline{CS}$ and WR are both active.

Command Register

The Command register provides direct control over each of the five general counters and controls access through the Data port by allowing the user to update the Data Pointer register. The "Command Description" section of this data sheet explains the detailed operation of each command. A summary of all commands appears in Figure 1-21. Six of the command types are used for direct software control of the counting process. Each of these six commands contains a 5-bit S field. In a linear-select fashion, each bit in the S field corresponds to one of the five general counters (S1 = Counter 1, S2 = Counter 2, etc.). When an S bit is a one, the specified operation is performed on the counter so designated; when an S bit is a zero, no operation occurs for the corresponding counter.

Data Pointer Register

The 6-bit Data Pointer register is loaded by issuing the appropriate command through the Control port to the Command register. As shown in Figure 1-8, the contents of the Data Pointer register are used to control the Data port multiplexer, selecting which internal register is to be accessible through the Data port.

The Data Pointer consists of a 3-bit Group Pointer, a 2-bit Element Pointer and a 1-bit Byte Pointer, depicted in Figure 1-9. The Byte Pointer bit indicates which byte of a 16-bit register is to be transferred on the next access through the Data port. Whenever the Data Pointer is loaded, the Byte Pointer bit is set to one, indicating a least-significant byte is expected. The Byte Pointer toggles following each 8-bit data transfer with an 8-bit data bus (MM13 = 0), or it always remains set with the 16-bit data bus option (MM13 = 1). The Element and Group pointers are used to select which internal register is to be accessible through the Data port. Although the contents of the Element and Group Pointer in the Data Pointer register cannot be read by the host processor, the Byte Pointer is available as a bit in the Status register.

Random access to any available internal data location can be accomplished by simply loading the Data Pointer using the command shown in Figure 1-10 and then initiating a data read or data write. This procedure can be used at any time, regardless of the setting of the Data Pointer Control bit (MM14). When the 8-bit data bus configuration is being used (MM13 = 0), two bytes of data would normally be transferred following the issuing of the "Load Data Pointer" command.

To permit the host processor to rapidly access the various internal registers, automatic sequencing of the Data Pointer is provided.

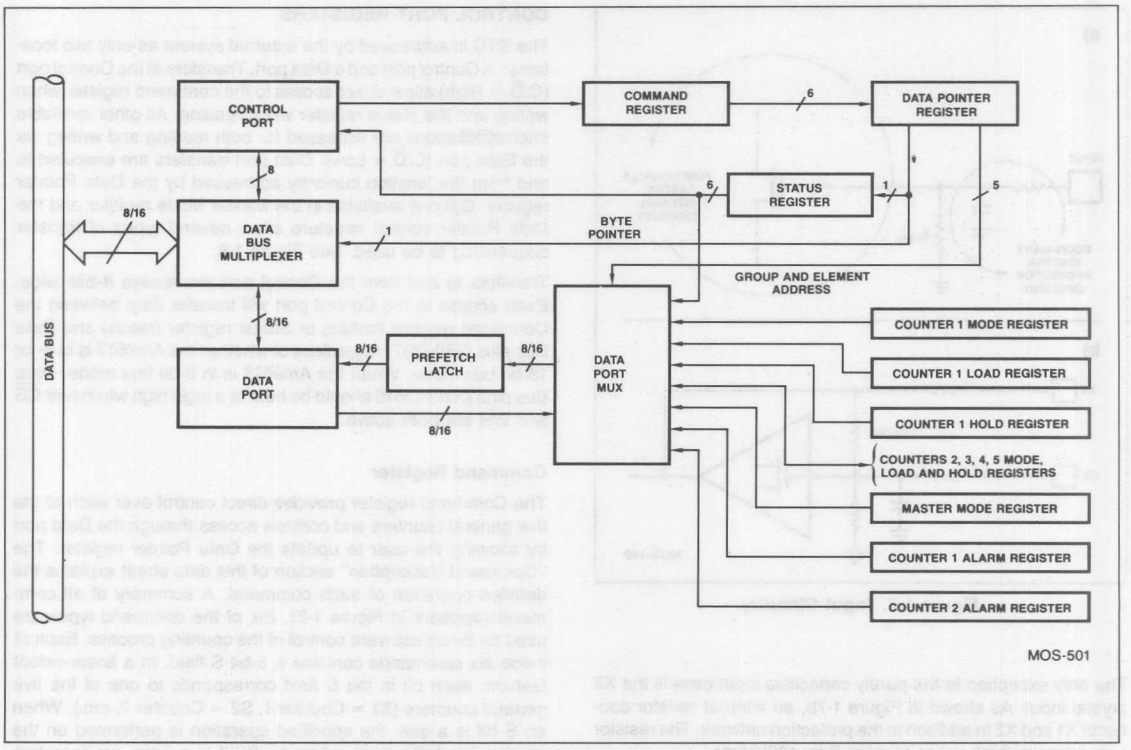


Figure 1-8. Am9513 Register Access

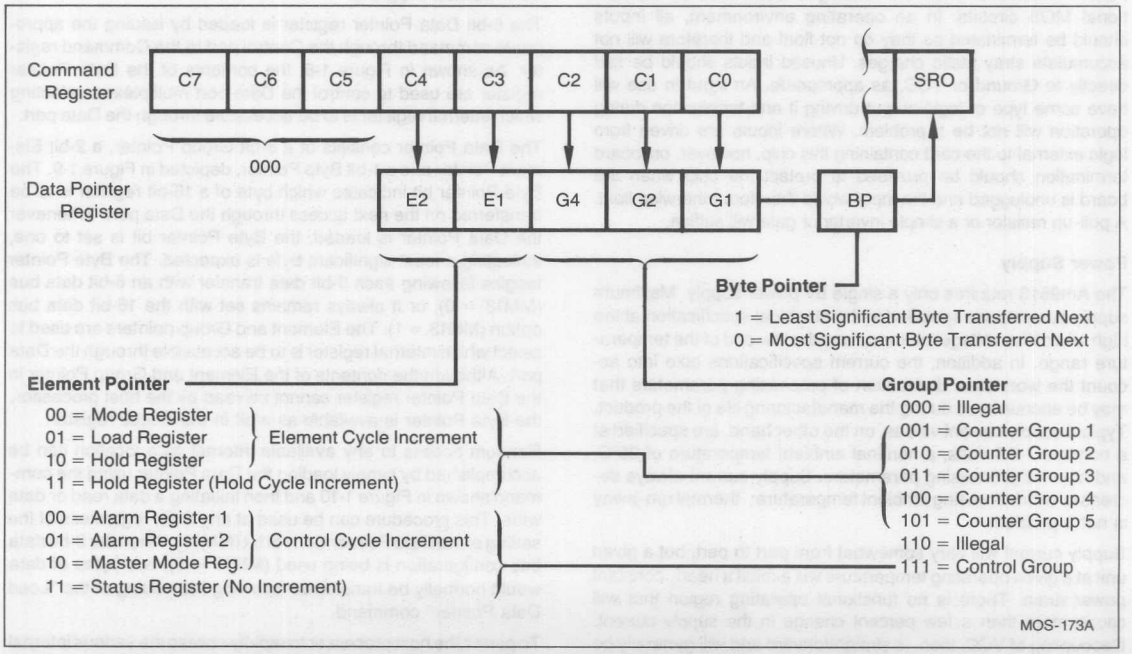


Figure 1-9. Data Pointer Register

	Element Cycle			Hold Cycle
	Mode Register	Load Register	Hold Register	Hold Register
Counter 1	FF01	FF09	FF11	FF19
Counter 2	FF02	FF0A	FF12	FF1A
Counter 3	FF03	FF0B	FF13	FF1B
Counter 4	FF04	FF0C	FF14	FF1C
Counter 5	FF05	FF0D	FF15	FF1D

Master Mode Register = FF17
Alarm 1 Register = FF07
Alarm 2 Register = FF0F

Notes:
1. All codes are in hex.
2. When used with an 8-bit bus, only the two low order hex digits should be written to the command port; the 'FF' prefix should be used only for a 16-bit data bus interface.

Figure 1-10. Load Data Pointer Commands

Sequencing is enabled by clearing Master Mode bit 14 (MM14) to zero. As shown in Figure 1-11, several types of sequencing are available depending on the data bus width being used and the initial Data Pointer value entered by command.

When E1 = 0 or E2 = 0 and G4, G2, G1 point to a Counter Group, the Data Pointer will proceed through the Element cycle. The Element field will automatically sequence through the three values 00, 01 and 10 starting with the value entered. When the transition from 10 to 00 occurs, the Group field will also be incremented by one. Note that the Element field in this case does not sequence to a value of 11. The Group field circulates only within the five Counter Group codes.

If E2, E1 = 11 and a Counter Group is selected, then only the Group field is sequenced. This is the Hold cycle. It allows the Hold registers to be sequentially accessed while bypassing the Mode and Load registers. The third type of sequencing is the Control cycle. If G4, G2, G1 = 111 and E2, E1 ≠ 11, the Element Pointer will be incremented through the values 00, 01 and 10, with no change to the Group Pointer.

When G4, G2, G1 = 111 and E2, E1 = 11, no incrementing takes place and only the Status register will be available through the Data port. Note that the Status register can also always be read directly through the Control port.

For all of these auto-sequence modes, if an 8-bit data bus is used, the Byte pointer will toggle after every data transfer to allow the least and most significant bytes to be transferred before the Element or Group Fields are incremented.

Prefetch Circuit

In order to minimize the read access time to internal Am9513 registers, a prefetch circuit is used for all read operations through the Data port. Following each read or write operation through the Data port, the Data Pointer register is updated to point to the next register to be accessed. Immediately following this update, the new register data is transferred to a special prefetch latch at the interface pad logic. When the user performs a subsequent read of the Data port, the data bus drivers are enabled, outputting the prefetched data on the bus. Since the internal data register is accessed prior to the start of the read operation, its access time is transparent to the user. In order to keep the prefetched data consistent with the Data Pointer, prefetches are also performed

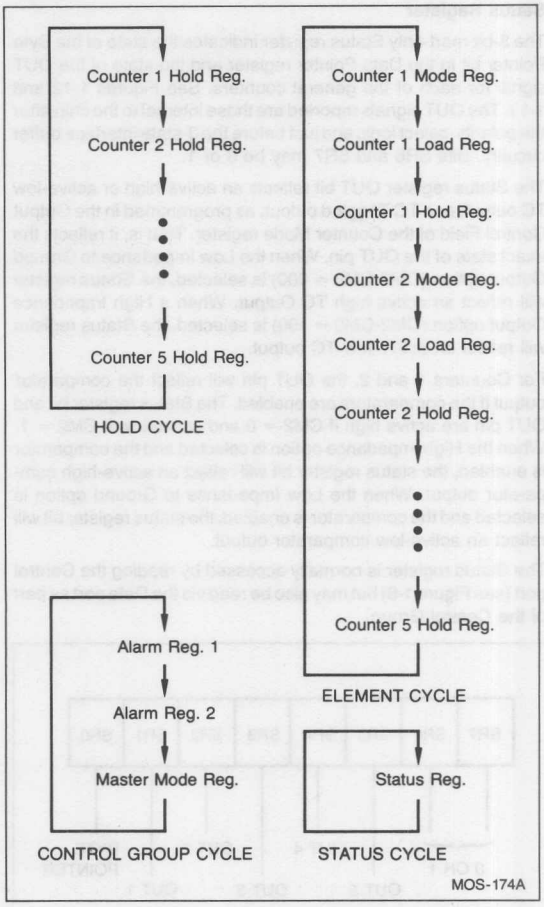


Figure 1-11. Data Pointer Sequencing

after each write to the Data port and after execution of the "Load Data Pointer" command. The following rules should be kept in mind regarding Data port Transfers.

1. The Data Pointer register should always be reloaded before reading from the Data port if a command other than "Load Data Pointer" was issued to the Am9513 following the last Data port read or write. The Data Pointer does not have to be loaded again if the first Data port transaction after a command entry is a write, since the Data port write will automatically cause a new prefetch to occur.
2. Operating modes N, O, Q, R and X allow the user to save the counter contents in the Hold register by applying an active-going gate edge. If the Data Pointer register had been pointing to the Hold register in question, the prefetched value will not correspond to the new value saved in the Hold register. To avoid reading an incorrect value, a new "Load Data Pointer" command should be issued before attempting to read the saved data. A Data port write (to another register) will also initiate a prefetch; subsequent reads will access the recently saved Hold register data. Many systems will use the "saving" gate edge to interrupt the host CPU. In systems such as this the interrupt service routine should issue a "Load Data Pointer" command prior to reading the saved data.

Status Register

The 8-bit read-only Status register indicates the state of the Byte Pointer bit in the Data Pointer register and the state of the OUT signal for each of the general counters. See Figures 1-12 and 1-19. The OUT signals reported are those internal to the chip after the polarity-select logic and just before the 3-state interface buffer circuitry. Bits SR6 and SR7 may be 0 or 1.

The Status register OUT bit reflects an active-high or active-low TC output, or a TC Toggled output, as programmed in the Output Control Field of the Counter Mode register. That is, it reflects the exact state of the OUT pin. When the Low Impedance to Ground Output option (CM2-CM0 = 000) is selected, the Status register will reflect an active-high TC Output. When a High Impedance Output option (CM2-CM0 = 100) is selected, the Status register will reflect an active-low TC output.

For Counters 1 and 2, the OUT pin will reflect the comparator output if the comparators are enabled. The Status register bit and OUT pin are active high if CM2 = 0 and active-low if CM2 = 1. When the High Impedance option is selected and the comparator is enabled, the status register bit will reflect an active-high comparator output. When the Low Impedance to Ground option is selected and the comparator is enabled, the status register bit will reflect an active-low comparator output.

The Status register is normally accessed by reading the Control port (see Figure 1-8) but may also be read via the Data port as part of the Control Group.

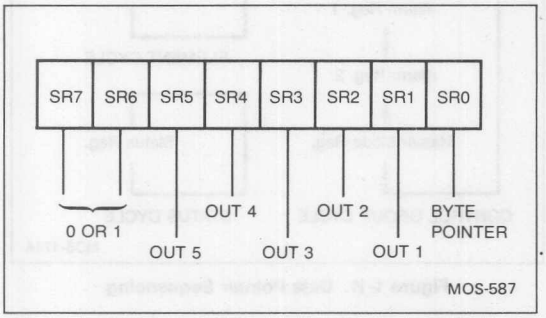


Figure 1-12. Status Register Bit Assignments

DATA PORT REGISTERS

Counter Logic Groups

As shown in Figures 1-2 and 1-3, each of the five Counter Logic Groups consists of a 16-bit general counter with associated control and output logic, a 16-bit Load register, a 16-bit Hold register and a 16-bit Mode register. In addition, Counter Groups 1 and 2 also include 16-bit Comparators and 16-bit Alarm registers. The comparator/alarm functions are controlled by the Master Mode register. The operation of the Counter Mode registers is the same for all five counters. The host CPU has both read and write access to all registers in the Counter Logic Groups through the Data port. The counter itself is never directly accessed.

Load Register

The 16-bit read/write Load register is used to control the effective length of the general counter. Any 16-bit value may be written into the Load register. That value can then be transferred into the counter each time the Terminal Count (TC) occurs. "Terminal Count" is defined as that period of time when the counter contents would have been zero if an external value had not been transferred into the counter. Thus, the terminal count frequency can be the input frequency divided by the value in the Load register. In all operating modes either the Load or Hold register will be transferred into the counter when TC occurs. In cases where values are being accumulated in the counter, the Load register action can become transparent by filling the Load register with all zeros.

Hold Register

The 16-bit read/write Hold register is dual-purpose. It can be used in the same way as the Load register, thus offering an alternate source for module definition for the counter. The Hold register may also be used to store accumulated counter values for later transfer to the host processor. This allows the count to be sampled while the counting process proceeds without interruption. Transfer of the counter contents into the Hold register is accomplished by the hardware interface in some operating modes or by software commands at any time.

Counter Mode Register

The 16-bit read/write Counter Mode register controls the gating, counting, output and source select functions within each Counter Logic Group. The "Counter Mode Control Options" section of this document describes the detailed control options available. Figure 1-18 shows the bit assignments for the Counter Mode registers.

Alarm Registers and Comparators

Added functions are available in the Counter Logic Groups for Counters 1 and 2 (see Figure 1-2). Each contains a 16-bit Alarm register and a 16-bit Comparator. When the value in the counter reaches the value in the Alarm register, the Comparator output will go true. The Master Mode register contains control bits to individually enable/disable the comparators. When enabled, the comparator output appears on the OUT pin of the associated counter in place of the normal counter output. The output will remain true as long as the comparison is true, that is, until the next input causes the count to change. The polarity of the Comparator output will be active-high if the Output Control field of the Counter Mode register is 001 or 010 and active-low if the Output Control field is 101.

MASTER MODE CONTROL OPTIONS

The 16-bit Master Mode (MM) register is used to control those internal activities that are not controlled by the individual Counter Mode registers. This includes frequency control, Time-of-Day operation, comparator controls, data bus width and data pointer sequencing. Figure 1-13 shows the bit assignments for the Master Mode register. This section describes the use of each control field.

Master Mode register bits MM12, MM13 and MM14 can be individually set and reset using commands issued to the Command register. In addition they can all be changed by writing directly to the Master Mode register.

After power-on reset or a Master Reset command, the Master Mode register is cleared to an all zero condition. This results in the following configuration:

Time-of-Day disabled
Both Comparators disabled
FOUT Source is frequency F1
FOUT Divider set for divide-by-16
FOUT gated on
Data Bus 8 bits wide
Data Pointer Sequencing enabled
Frequency Scaler divides in binary

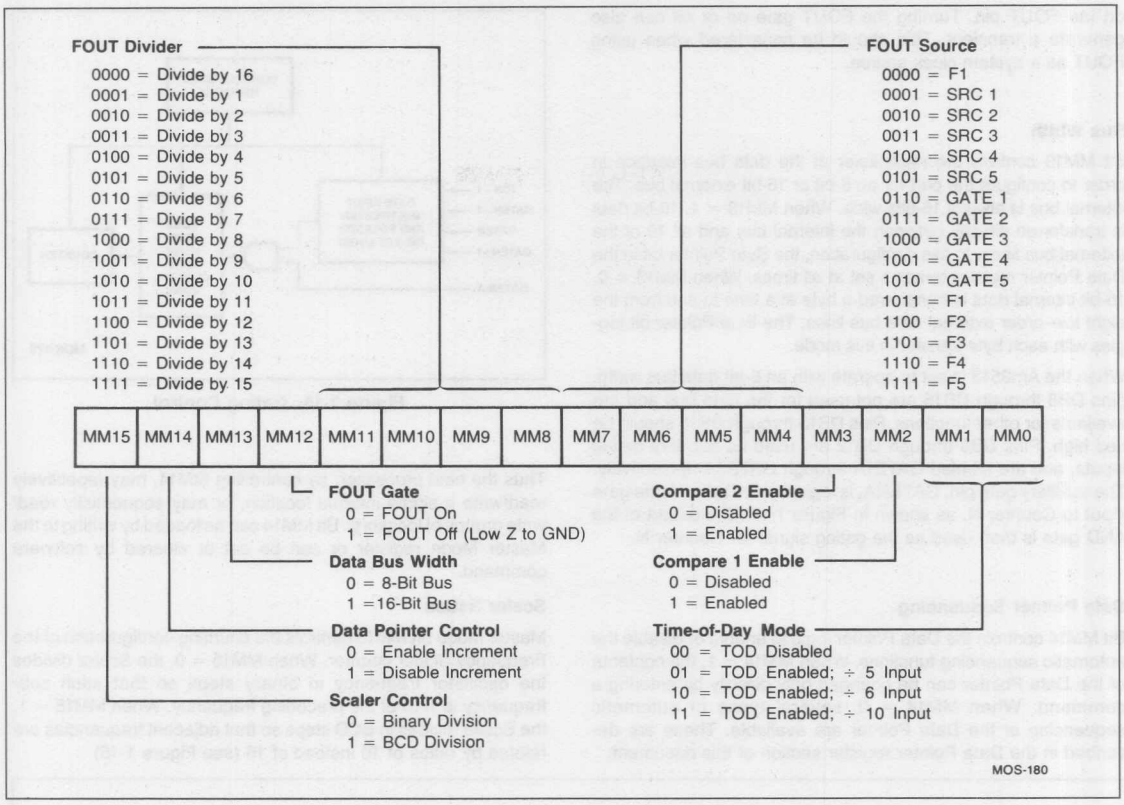


Figure 1-13. Master Mode Register Bit Assignments

Time-of-Day

Bits MM0 and MM1 of the Master Mode register specify the Time-of-Day (TOD) options. When MM0 = 0 and MM1 = 0, the special logic used to implement TOD is disabled and Counters 1 and 2 will operate in exactly the same way as Counters 3, 4 and 5. When MM0 = 1 or MM1 = 1, additional counter decoding and control logic is enabled on Counters 1 and 2 which causes their decades to turn over at the counts that generate appropriate 24-hour TOD accumulations. For additional information, see the Time-of-Day chapter in this applications note.

Comparator Enable

Bits MM2 and MM3 control the Comparators associated with Counter 1 and 2. When a Comparator is enabled, its output is substituted for the normal counter output on the associated OUT1 or OUT2 pin. The comparator output will be active-high if the output control field of the Counter Mode register is 001 or 010 and active low for a code of 101. Once the compare output is true, it will remain so until the count changes and the comparison therefore goes false.

The two Comparators can always be used individually in any operating mode. One special case occurs when the Time-of-Day option is invoked and both Comparators are enabled. The operation of Comparator 2 will then be conditioned by Comparator 1 so that a full 32-bit compare must be true in order to generate a true signal on OUT2. OUT1 will continue, as usual, to reflect the state of the 16-bit comparison between Alarm 1 and Counter 1.

FOUT Source

Master Mode bits MM4 through MM7 specify the source input for the FOUT divider. Fifteen inputs are available for selection and they include the five Source pins, the five Gate pins and the five internal frequencies derived from the oscillator. The 16th combination of the four control bits (all zeros) is used to assure that an active frequency is available at the input to the FOUT divider following reset.

FOUT Divider

Bits MM8 through MM11 specify the dividing ratio for the FOUT Divider. The FOUT source (selected by bits MM4 through MM7) is divided by an integer value between 1 and 16, inclusive, and is then passed to the FOUT output buffer. After power-on or reset, the FOUT divider is set to divide-by-16.

FOUT Gate

Master Mode bit MM12 provides a software gating capability for the FOUT signal. When MM12 = 1, FOUT is off and in a low impedance state to ground. MM12 may be set or cleared in conjunction with the loading of the other bits in the Master Mode register; alternatively, there are commands that allow MM12 to be individually set or cleared directly without changing any other Master Mode bits. After power-up or reset, FOUT is gated on.

When changing the FOUT divider ratio or FOUT source, transient pulses as short as half the period of the FOUT source may appear

on the FOUT pin. Turning the FOUT gate on or off can also generate a transient. This should be considered when using FOUT as a system clock source.

Bus Width

Bit MM13 controls the multiplexer at the data bus interface in order to configure the part for an 8-bit or 16-bit external bus. The internal bus is always 16-bits wide. When MM13 = 1, 16-bit data is transferred directly between the internal bus and all 16 of the external bus lines. In this configuration, the Byte Pointer bit in the Data Pointer register remains set at all times. When MM13 = 0, 16-bit internal data is transferred a byte at a time to and from the eight low-order external data bus lines. The Byte Pointer bit toggles with each byte transfer in this mode.

When the Am9513 is set to operate with an 8-bit data bus width, pins DB8 through DB15 are not used for the data bus and are available for other functions. Pins DB13 through DB15 should be tied high. Pins DB8 through DB12 are used as auxiliary gating inputs, and are labeled GATE1A through GATE5A respectively. The auxiliary gate pin, GATENA, is logically ANDed with the gate input to Counter N, as shown in Figure 1-14. The output of the AND gate is then used as the gating signal for Counter N.

Data Pointer Sequencing

Bit MM14 controls the Data Pointer logic to enable or disable the automatic sequencing functions. When MM14 = 1, the contents of the Data Pointer can be changed only directly by entering a command. When MM14 = 0, several types of automatic sequencing of the Data Pointer are available. These are described in the Data Pointer register section of this document.

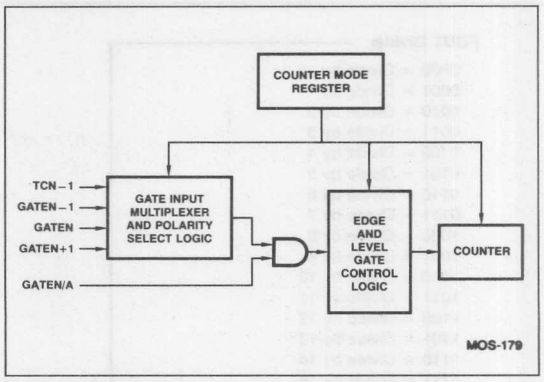
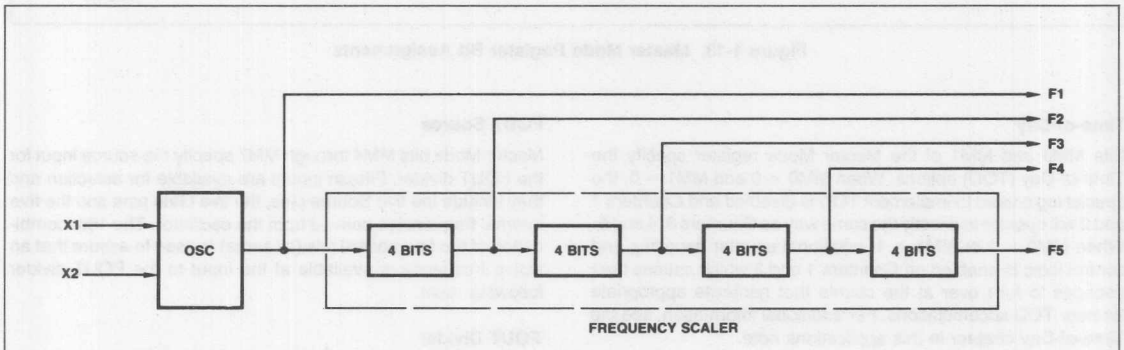


Figure 1-14. Gating Control

Thus the host processor, by controlling MM14, may repetitively read/write a single internal location, or may sequentially read/write groups of locations. Bit MM14 can be loaded by writing to the Master Mode register or can be set or cleared by software command.

Scaler Ratios

Master Mode bit MM15 controls the counting configuration of the Frequency Scaler counter. When MM15 = 0, the Scaler divides the oscillator frequency in binary steps so that each subfrequency is 1/16 of the preceding frequency. When MM15 = 1, the Scaler divides in BCD steps so that adjacent frequencies are related by ratios of 10 instead of 16 (see Figure 1-15).



Frequency	BCD Scaling MM15 = 1	Binary Scaling MM15 = 0
F1	OSC	OSC
F2	F1 ÷ 10	F1 ÷ 16
F3	F1 ÷ 100	F1 ÷ 256
F4	F1 ÷ 1,000	F1 ÷ 4,096
F5	F1 ÷ 10,000	F1 ÷ 65,536

MOS-150

Figure 1-15. Frequency Scaler Ratios

Counter Mode	A	B	C	D	E	F	G	H	I	J	K	L
Special Gate (CM7)	0	0	0	0	0	0	0	0	0	0	0	0
Reload Source (CM6)	0	0	0	0	0	0	1	1	1	1	1	1
Repetition (CM5)	0	0	0	1	1	1	0	0	0	1	1	1
Gate Control (CM15-CM13)	000	LEVEL	EDGE	000	LEVEL	EDGE	000	LEVEL	EDGE	000	LEVEL	EDGE
Count to TC once, then disarm	X	X	X									
Count to TC twice, then disarm							X	X	X			
Count to TC repeatedly without disarming				X	X	X				X	X	X
Gate input does not gate counter input	X			X			X			X		
Count only during active gate level		X			X			X			X	
Start count on active gate edge and stop count on next TC			X			X						
Start count on active gate edge and stop count on second TC									X			X
No hardware retriggering	X	X	X	X	X	X	X	X	X	X	X	X
Reload counter from Load Register on TC	X	X	X	X	X	X						
Reload counter on each TC, alternating reload source between Load and Hold Registers							X	X	X	X	X	X
Transfer Load Register into counter on each TC that gate is LOW, transfer Hold Register into counter on each TC that gate is HIGH.												
On active gate edge transfer counter into Hold Register and then reload counter from Load Register												

Counter Mode	M	N	O	P	Q	R	S	T	U	V	W	X
Special Gate (CM7)	1	1	1	1	1	1	1	1	1	1	1	1
Reload Source (CM6)	0	0	0	0	0	0	1	1	1	1	1	1
Repetition (CM5)	0	0	0	1	1	1	0	0	0	1	1	1
Gate Control (CM15-CM13)	000	LEVEL	EDGE	000	LEVEL	EDGE	000	LEVEL	EDGE	000	LEVEL	EDGE
Count to TC once, then disarm		X	X									
Count to TC twice, then disarm							X					
Count to TC repeatedly without disarming					X	X				X		X
Gate input does not gate counter input							X			X		
Count only during active gate level		X			X							
Start count on active gate edge and stop count on next TC			X			X						X
Start count on active gate edge and stop count on second TC												
No hardware retriggering							X			X		X
Reload counter from Load Register on TC		X	X		X	X						X
Reload counter on each TC, alternating reload source between Load and Hold Registers.												
Transfer Load Register into counter on each TC that gate is LOW, transfer Hold Register into counter on each TC that gate is HIGH.							X			X		
On active gate edge transfer counter into Hold Register and then reload counter from Load Register		X	X		X	X						
On active gate edge transfer counter into Hold Register, but counting continues												X

Note: Counter modes M, P, T, U and W are reserved and should not be used.

Figure 1-16. Counter Mode Operating Summary

COUNTER MODE DESCRIPTIONS

Counter Mode register bits CM15-CM13 and CM7-CM5 select the operating mode for each counter (see Figure 1-16). To simplify references to a particular mode, each mode is assigned a letter from A through X. Representative waveforms for the counter modes are illustrated in Figures 1-17a through 1-17v. (Because the letter suffix in the figure number is keyed to the mode, Figures 1-17m, 1-17p, 1-17t, 1-17u and 1-17w do not exist.) The figures assume down counting on rising source edges. Those modes which automatically disarm the counter (CM5 = 0) are shown with the $\overline{WR}$ pulse entering the required ARM command; for modes which count repetitively (CM5 = 1) the ARM command is omitted. The retriggering modes (N, O, Q and R) are shown with one retrigger operation. Both a TC output waveform and a TC Toggled output waveform are shown for each mode. The symbols L and H are used to represent count values equal to the Load and Hold register contents, respectively. The symbols K and N represent arbitrary count values. For each mode, the required bit pattern in the Counter Mode register is shown; "don't care" bits are marked "X." These figures are designed to clarify the mode descriptions; the Am9513 Electrical Specification should be used as the authoritative reference for timing relationships between signals. Appendix B provides a key to the waveform symbols used in these diagrams.

To keep the following mode descriptions concise and to the point, the phrase "source edges" is used to refer to active-going source edges only, not to inactive-going edges. Similarly, the phrase "gate edges" refers only to active-going gate edges. Also, again to avoid verbosity and euphuism, the descriptions of some modes state that a counter is stopped or disarmed "on a TC, inhibiting further counting." As is fully explained in the TC section of this document, for these modes the counter is actually stopped or disarmed following the active-going source edge which drives the counter out of TC. In other words, since a counter in the TC state always counts, irrespective of its gating or arming status, the stopping or disarming of the count sequence is delayed until TC is terminated.

MODE A
Software-Triggered Strobe with No Hardware Gating

CM15	CM14	CM13	CM12	CM11	CM10	CM9	CM8
0	0	0	X	X	X	X	X

CM7	CM6	CM5	CM4	CM3	CM2	CM1	CM0
0	0	0	X	X	X	X	X

Mode A, shown in Figure 1-17a, is one of the simplest operating modes. The counter will be available for counting source edges when it is issued an ARM command. On each TC the counter will reload from the Load register and automatically disarm itself, inhibiting further counting. Counting will resume when a new ARM command is issued.

MODE B
Software-Triggered Strobe with Level Gating

CM15	CM14	CM13	CM12	CM11	CM10	CM9	CM8
LEVEL			X	X	X	X	X

CM7	CM6	CM5	CM4	CM3	CM2	CM1	CM0
0	0	0	X	X	X	X	X

Mode B, shown in Figure 1-17b, is identical to Mode A except that source edges are counted only when the assigned Gate is active. The counter must be armed before counting can occur. Once armed, the counter will count all source edges which occur while the Gate is active and disregard those edges which occur while the Gate is inactive. This permits the Gate to turn the count process on and off. On each TC the counter will reload from the Load register and automatically disarm itself, inhibiting further counting until a new ARM command is issued.

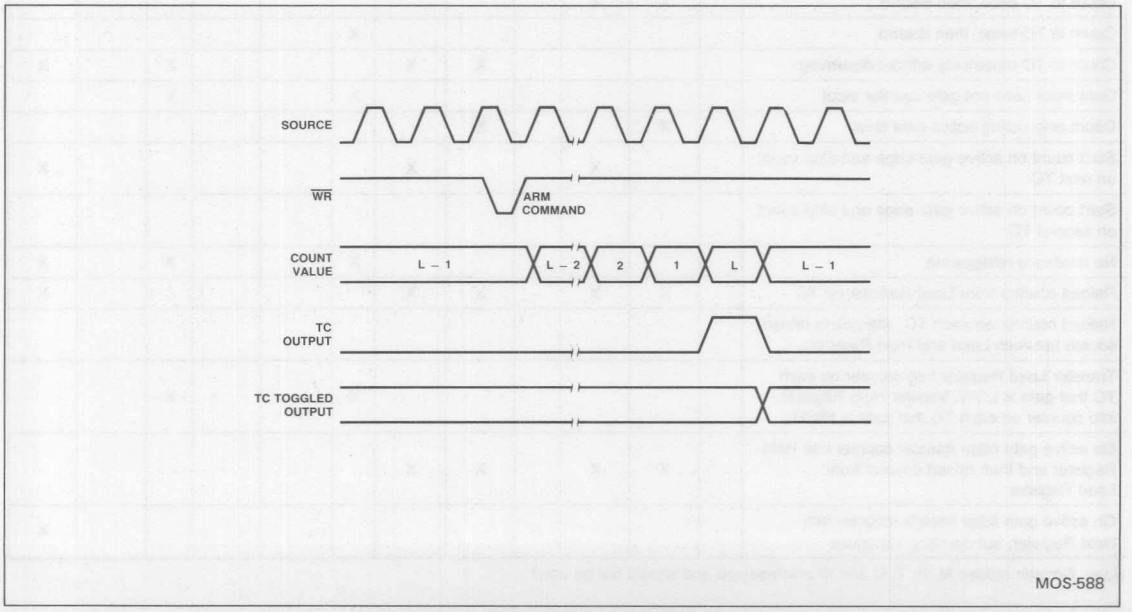


Figure 1-17a. Mode A Waveforms

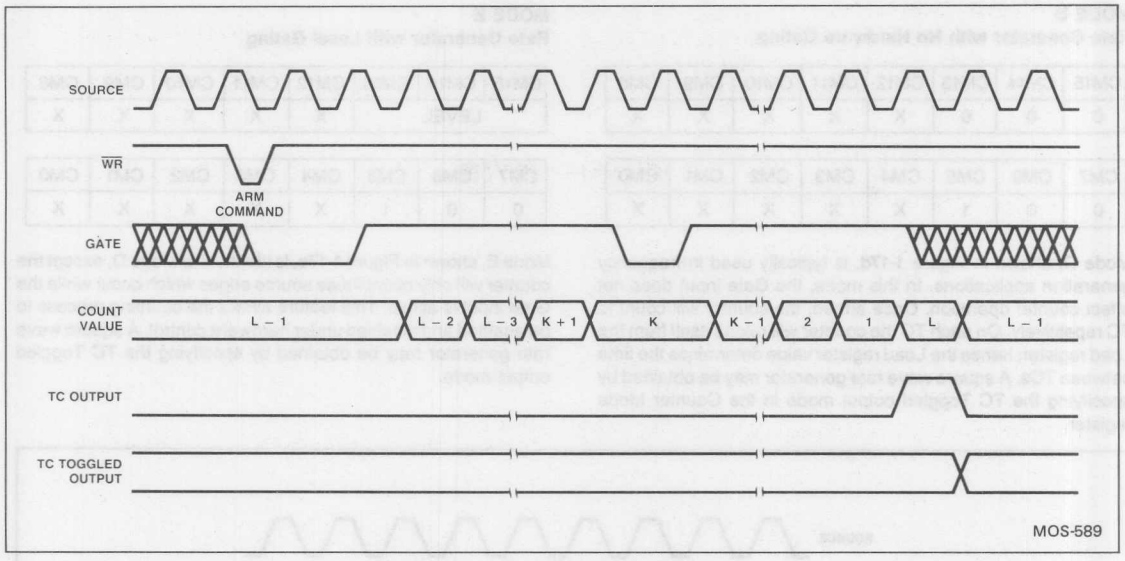


Figure 1-17b. Mode B Waveforms

MODE C
Hardware-Triggered Strobe

CM15	CM14	CM13	CM12	CM11	CM10	CM9	CM8
EDGE			X	X	X	X	X

CM7	CM6	CM5	CM4	CM3	CM2	CM1	CM0
0	0	0	X	X	X	X	X

Mode C, shown in Figure 1-17c, is identical to Mode A, except that counting will not begin until a Gate edge is applied to the armed counter. The counter must be armed before application of the triggering Gate edge; Gate edges applied to a disarmed counter are disregarded. The counter will start counting on the first source edge after the triggering Gate edge and will continue counting until TC. At TC, the counter will reload from the Load register and automatically disarm itself. Counting will then remain inhibited until a new ARM command and a new Gate edge are applied in that order. Note that after application of a triggering Gate edge, the Gate input will be disregarded for the remainder of the count cycle. This differs from Mode B, where the Gate can be modulated throughout the count cycle to stop and start the counter.

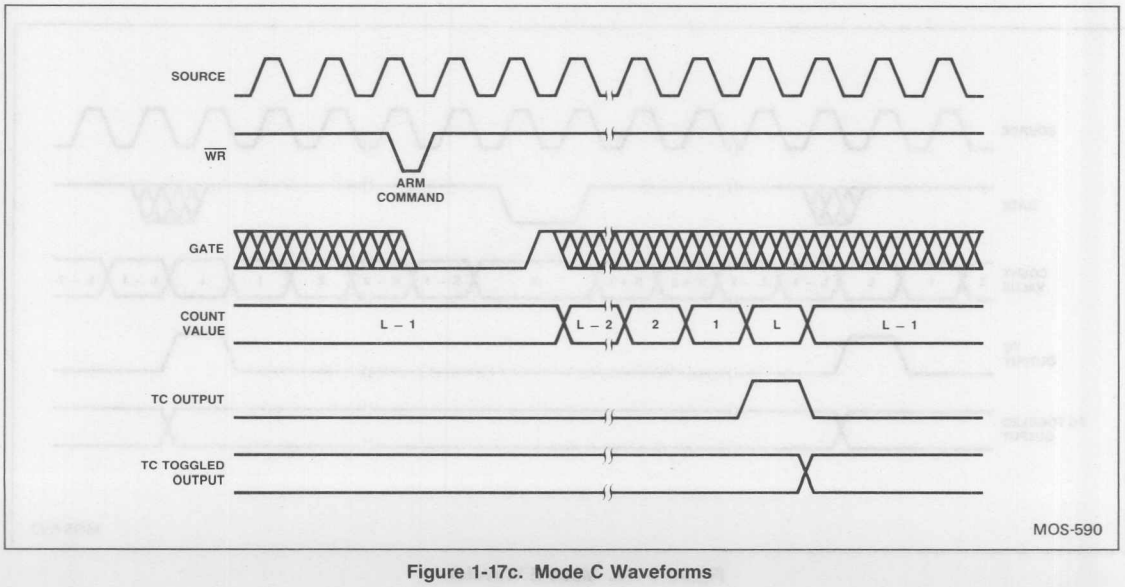


Figure 1-17c. Mode C Waveforms

MODE D
Rate Generator with No Hardware Gating

CM15	CM14	CM13	CM12	CM11	CM10	CM9	CM8
0	0	0	X	X	X	X	X

CM7	CM6	CM5	CM4	CM3	CM2	CM1	CM0
0	0	1	X	X	X	X	X

Mode D, shown in Figure 1-17d, is typically used in frequency generation applications. In this mode, the Gate input does not affect counter operation. Once armed, the counter will count to TC repetitively. On each TC the counter will reload itself from the Load register; hence the Load register value determines the time between TCs. A square wave rate generator may be obtained by specifying the TC Toggled output mode in the Counter Mode register.

MODE E
Rate Generator with Level Gating

CM15	CM14	CM13	CM12	CM11	CM10	CM9	CM8
LEVEL			X	X	X	X	X

CM7	CM6	CM5	CM4	CM3	CM2	CM1	CM0
0	0	1	X	X	X	X	X

Mode E, shown in Figure 1-17e, is identical to Mode D, except the counter will only count those source edges which occur while the Gate input is active. This feature allows the counting process to be enabled and disabled under hardware control. A square wave rate generator may be obtained by specifying the TC Toggled output mode.

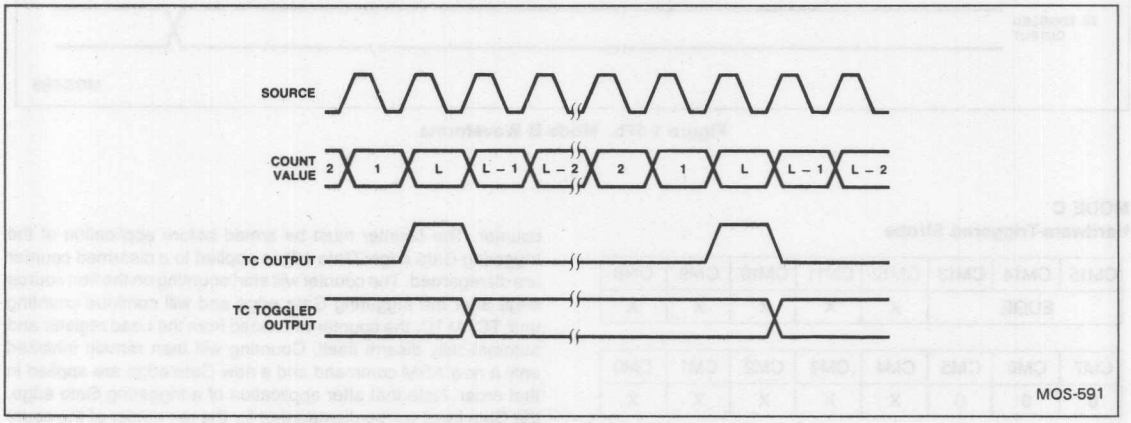


Figure 1-17d. Mode D Waveforms

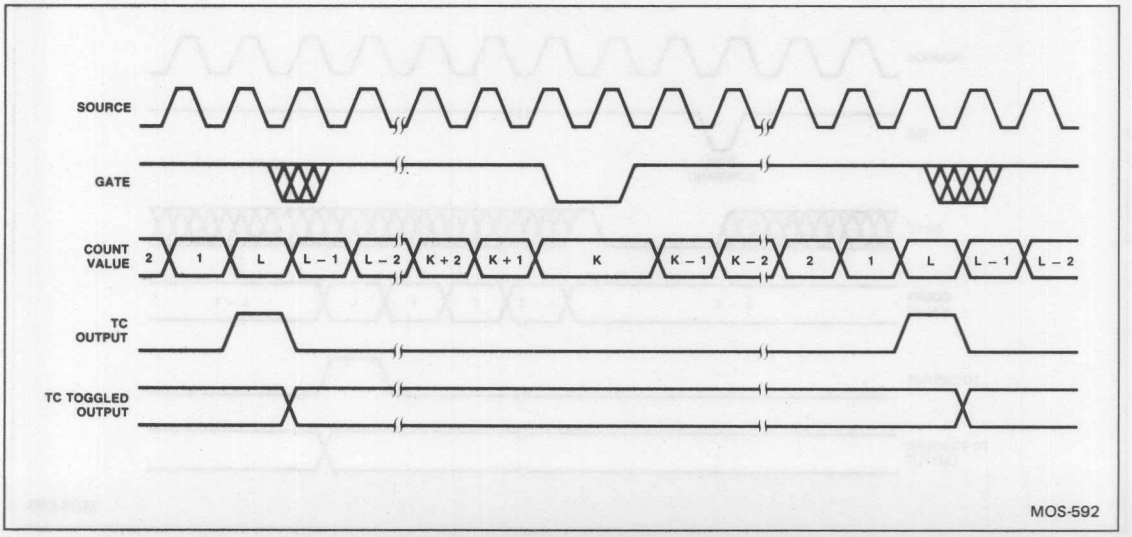


Figure 1-17e. Mode E Waveforms

MODE F
Non-Retriggerable One-Shot

CM15	CM14	CM13	CM12	CM11	CM10	CM9	CM8
EDGE			X	X	X	X	X

CM7	CM6	CM5	CM4	CM3	CM2	CM1	CM0
0	0	1	X	X	X	X	X

Mode F, shown in Figure 1-17f, provides a non-retriggerable one-shot timing function. The counter must be armed before it will function. Application of a Gate edge to the armed counter will enable counting. When the counter reaches TC, it will reload itself from the Load register. The counter will then stop counting, awaiting a new Gate edge. Note that unlike Mode C, a new ARM command is not needed after TC, only a new Gate edge. After application of a triggering Gate edge, the Gate input is disregarded until TC.

MODE G
Software-Triggered Delayed Pulse One-Shot

CM15	CM14	CM13	CM12	CM11	CM10	CM9	CM8
0	0	0	X	X	X	X	X

CM7	CM6	CM5	CM4	CM3	CM2	CM1	CM0
0	1	0	X	X	X	X	X

In Mode G, the Gate does not affect the counter's operation. Once armed, the counter will count to TC twice and then automatically disarm itself. For most applications, the counter will initially be loaded from the Load register either by a LOAD command or by the last TC of an earlier timing cycle. Upon counting to the first TC, the counter will reload itself from the Hold register. Counting will proceed until the second TC, when the counter will reload itself from the Load register and automatically disarm itself, inhibiting further counting. Counting can be resumed by issuing a new ARM command. A software-triggered delayed pulse one-shot may be generated by specifying the TC Toggled output mode in the Counter Mode register. The initial counter contents control the delay from the ARM command until the output pulse starts. The Hold register contents control the pulse duration. Mode G is shown in Figure 1-17g.

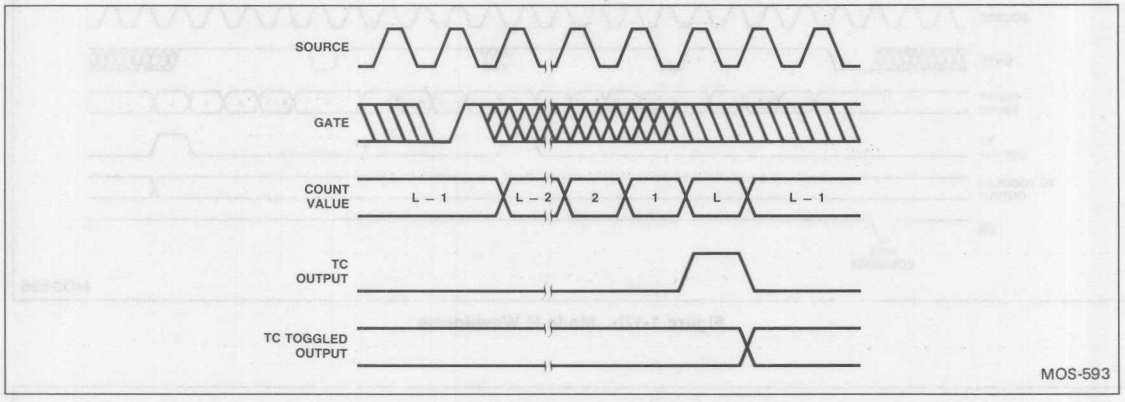


Figure 1-17f. Mode F Waveforms

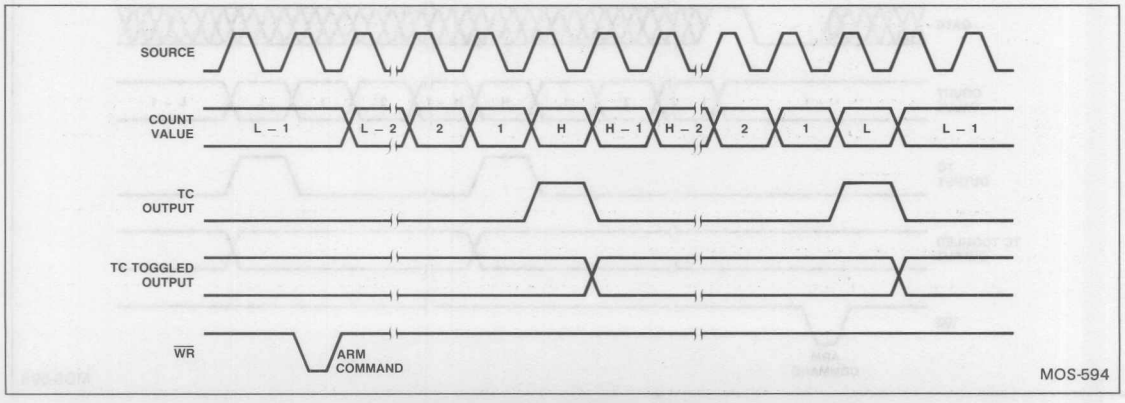


Figure 1-17g. Mode G Waveforms

MODE H
Software-Triggered Delayed Pulse One-Shot with Hardware Gating

CM15	CM14	CM13	CM12	CM11	CM10	CM9	CM8
LEVEL			X	X	X	X	X

CM7	CM6	CM5	CM4	CM3	CM2	CM1	CM0
0	1	0	X	X	X	X	X

Mode H, shown in Figure 1-17h, is identical to Mode G except that the Gate input is used to qualify which source edges are to be counted. The counter must be armed for counting to occur. Once armed, the counter will count all source edges that occur while the Gate is active and disregard those source edges that occur while the Gate is inactive. This permits the Gate to turn the count process on and off. As with Mode G, the counter will be reloaded from the Hold register on the first TC and reloaded from the Load register and disarmed on the second TC. This mode allows the Gate to control the extension of both the initial output delay time and the pulse width.

MODE I
Hardware-Triggered Delayed Pulse Strobe

CM15	CM14	CM13	CM12	CM11	CM10	CM9	CM8
EDGE			X	X	X	X	X

CM7	CM6	CM5	CM4	CM3	CM2	CM1	CM0
0	1	0	X	X	X	X	X

Mode I, shown in Figure 1-17i, is identical to Mode G, except that counting will not begin until a Gate edge is applied to an armed counter. The counter must be armed before application of the triggering Gate edge; Gate edges applied to a disarmed counter are disregarded. An armed counter will start counting on the first source edge after the triggering Gate edge. Counting will then proceed in the same manner as in Mode G. After the second TC, the counter will disarm itself. An ARM command and Gate edge must be issued in this order to restart counting. Note that after application of a triggering Gate edge, the Gate input will be disregarded until the second TC. This differs from Mode H, where the Gate can be modulated throughout the count cycle to stop and start the counter.

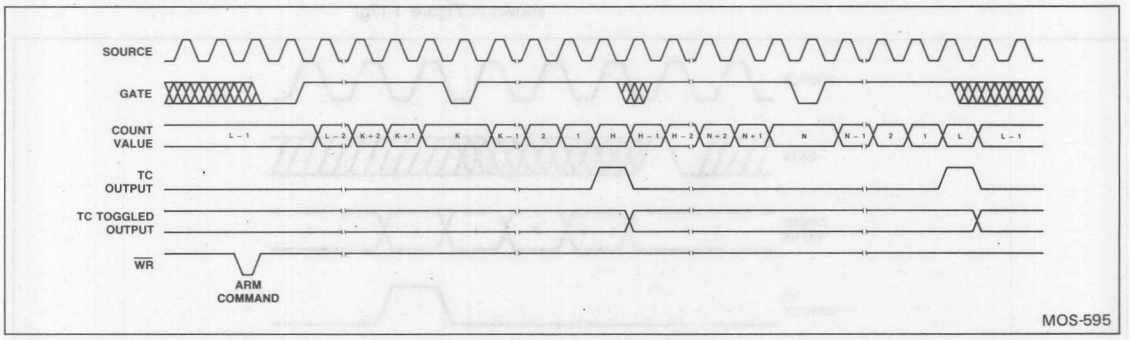


Figure 1-17h. Mode H Waveforms

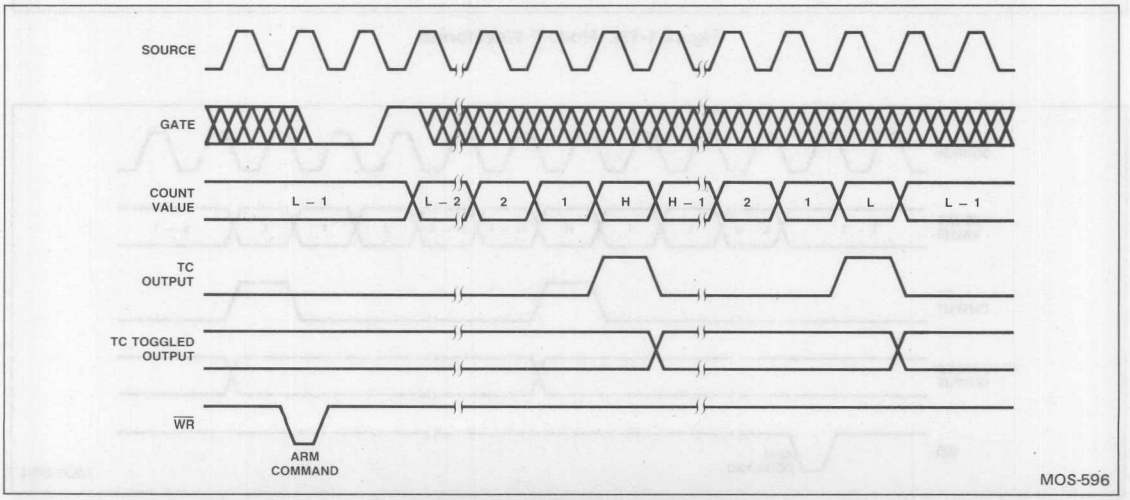


Figure 1-17i. Mode I Waveforms

MODE J
Variable Duty Cycle Rate Generator with No Hardware Gating

CM15	CM14	CM13	CM12	CM11	CM10	CM9	CM8
0	0	0	X	X	X	X	X

CM7	CM6	CM5	CM4	CM3	CM2	CM1	CM0
0	1	1	X	X	X	X	X

Mode J, shown in Figure 1-17j, will find the greatest usage in frequency generation applications with variable duty cycle requirements. Once armed, the counter will count continuously until it is issued a DISARM command. On the first TC, the counter will be reloaded from the Hold register. Counting will then proceed until the second TC at which time the counter will be reloaded from the Load register. Counting will continue, with the reload source alternating on each TC, until a DISARM command is issued to the counter. (The third TC reloads from the Hold register, the fourth TC reloads from the Load register, etc.) A variable duty cycle output can be generated by specifying the TC Toggled output in the Counter Mode register. The Load and Hold values then directly control the output duty cycle, with high resolution available when relatively high count values are used.

MODE K
Variable Duty Cycle Rate Generator with Level Gating

CM15	CM14	CM13	CM12	CM11	CM10	CM9	CM8
LEVEL			X	X	X	X	X

CM7	CM6	CM5	CM4	CM3	CM2	CM1	CM0
0	1	1	X	X	X	X	X

Mode K, shown in Figure 1-17k, is identical to Mode J except that source edges are only counted when the Gate is active. The counter must be armed for counting to occur. Once armed, the counter will count all source edges which occur while the Gate is active and disregard those source edges which occur while the Gate is inactive. This permits the Gate to turn the count process on and off. As with Mode J, the reload source used will alternate on each TC, starting with the Hold register on the first TC after any ARM command. When the TC Toggled output is used, this mode allows the Gate to modulate the duty cycle of the output waveform. It can affect both the high and low portions of the output waveform.

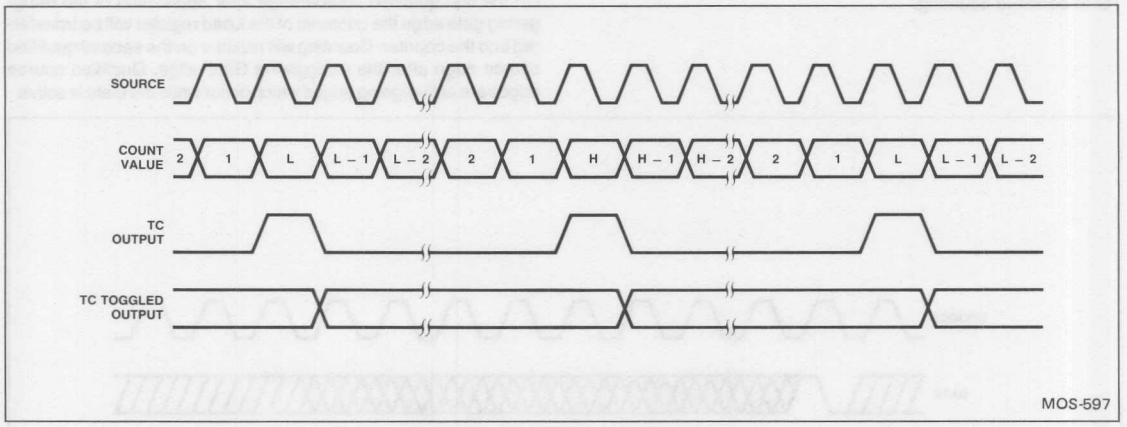


Figure 1-17j. Mode J Waveforms

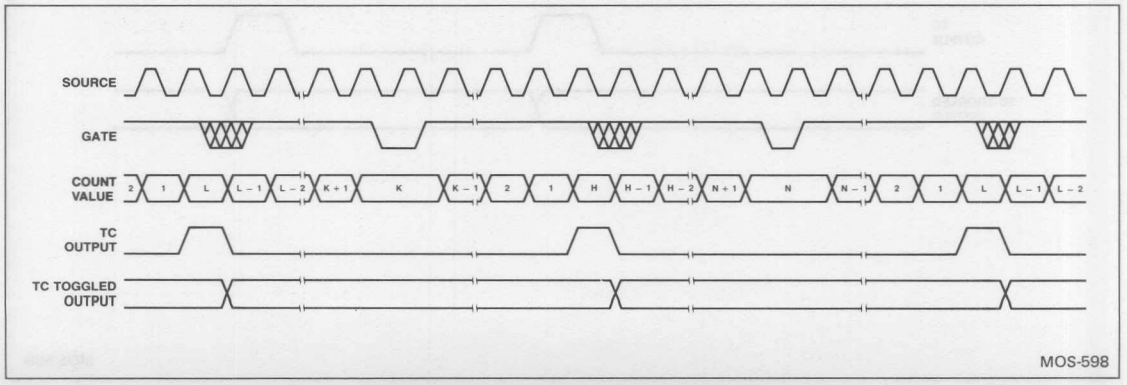


Figure 1-17k. Mode K Waveforms

MODE L
Hardware-Triggered Delayed Pulse One-Shot

CM15	CM14	CM13	CM12	CM11	CM10	CM9	CM8
EDGE			X	X	X	X	X

CM7	CM6	CM5	CM4	CM3	CM2	CM1	CM0
0	1	1	X	X	X	X	X

Mode L, shown in Figure 1-17l, is similar to Mode J except that counting will not begin until a Gate edge is applied to an armed counter. The counter must be armed before application of the triggering Gate edge; Gate edges applied to a disarmed counter are disregarded. The counter will start counting source edges after the triggering Gate edge and counting will proceed until the second TC. Note that after application of a triggering Gate edge, the Gate input will be disregarded for the remainder of the count cycle. This differs from Mode K, where the gate can be modulated throughout the count cycle to stop and start the counter. On the first TC after application of the triggering Gate edge, the counter will be reloaded from the Hold register. On the second TC, the counter will be reloaded from the Load register and counting will stop until a new gate edge is issued to the counter. Note that unlike Mode K, new Gate edges are required after every second TC to continue counting.

MODE N
Software-Triggered Strobe with Level Gating and Hardware Retriggering

CM15	CM14	CM13	CM12	CM11	CM10	CM9	CM8
LEVEL			X	X	X	X	X

CM7	CM6	CM5	CM4	CM3	CM2	CM1	CM0
1	0	0	X	X	X	X	X

Mode N, shown in Figure 1-17n, provides a software-triggered strobe with level gating that is also hardware retriggerable. The counter must first be issued an ARM command before counting can occur. Once armed, the counter will count all source edges which occur while the gate is active and disregard those source edges which occur while the Gate is inactive. This permits the Gate to turn the count process on and off. After the issuance of an ARM command and the application of an active Gate, the counter will count to TC. Upon reaching TC, the counter will reload from the Load register and automatically disarm itself, inhibiting further counting. Counting will resume upon the issuance of a new ARM command. All active-going Gate edges issued to an armed counter will cause a retrigger operation. Upon application of the Gate edge, the counter contents will be saved in the Hold register. On the first qualified source edge after application of the retriggering gate edge the contents of the Load register will be transferred into the counter. Counting will resume on the second qualified source edge after the retriggering Gate edge. Qualified source edges are active-going edges which occur while the Gate is active.

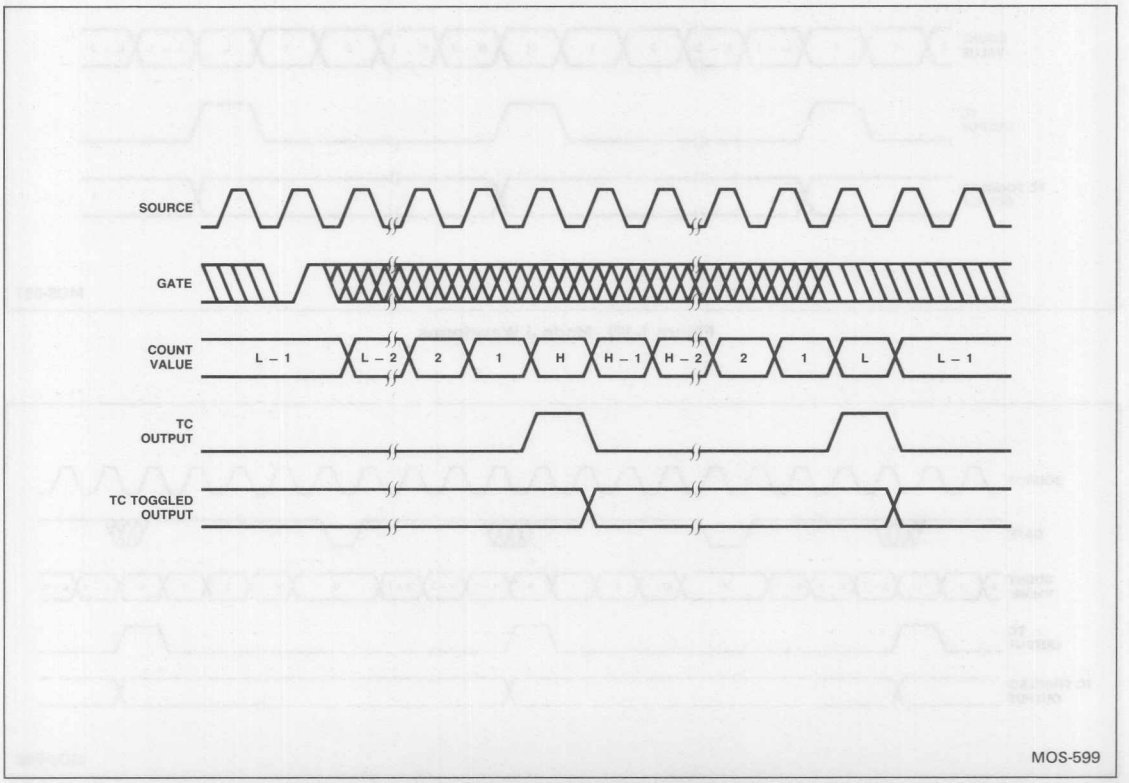


Figure 1-17l. Mode L Waveforms

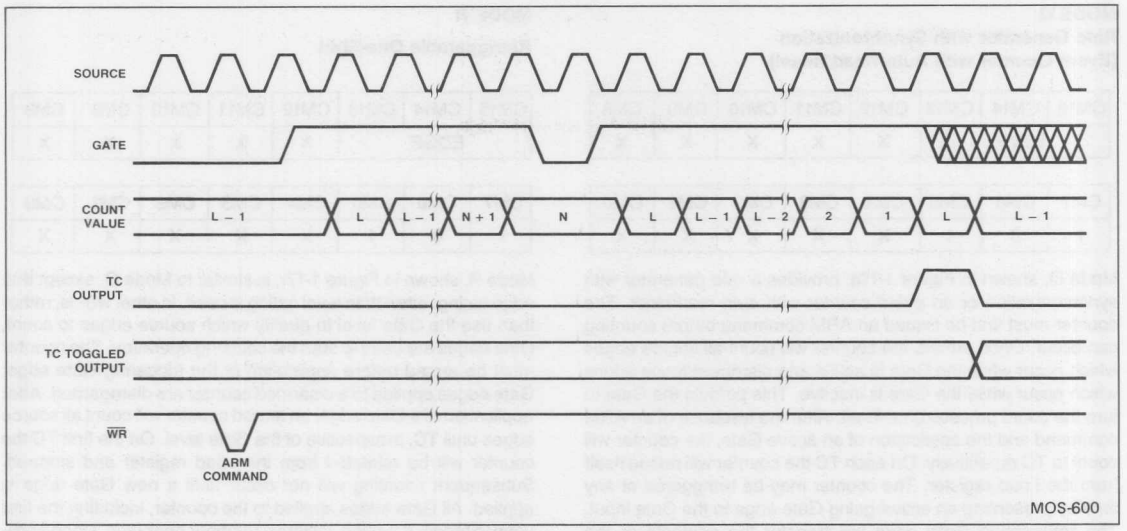


Figure 1-17n. Mode N Waveforms

MODE O
Software-Triggered Strobe with Edge Gating and Hardware Retriggering

CM15	CM14	CM13	CM12	CM11	CM10	CM9	CM8
EDGE			X	X	X	X	X

CM7	CM6	CM5	CM4	CM3	CM2	CM1	CM0
1	0	0	X	X	X	X	X

Mode O, shown in Figure 1-17o, is similar to Mode N, except that counting will not begin until an active-going Gate edge is applied to an armed counter and the Gate level is not used to modulate counting. The counter must be armed before application of the triggering Gate edge; Gate edges applied to a disarmed counter are disregarded. Irrespective of the Gate level, the counter will count all source edges after the triggering Gate edge until the first TC. On the first TC the counter will be reloaded from the Load register and disarmed. A new ARM command and a new Gate edge must be applied in that order to initiate a new counting cycle. Unlike Modes C, F, I and L, which disregard the Gate input once counting starts, in Mode O the count process will be retriggered on all active-going Gate edges, including the first Gate edge used to start the counter. On each retriggering Gate edge, the counter contents will be transferred into the Hold register. On the first source edge after the retriggering Gate edge the Load register contents will be transferred into the counter. Counting will resume on the second-source edge after a retrigger.

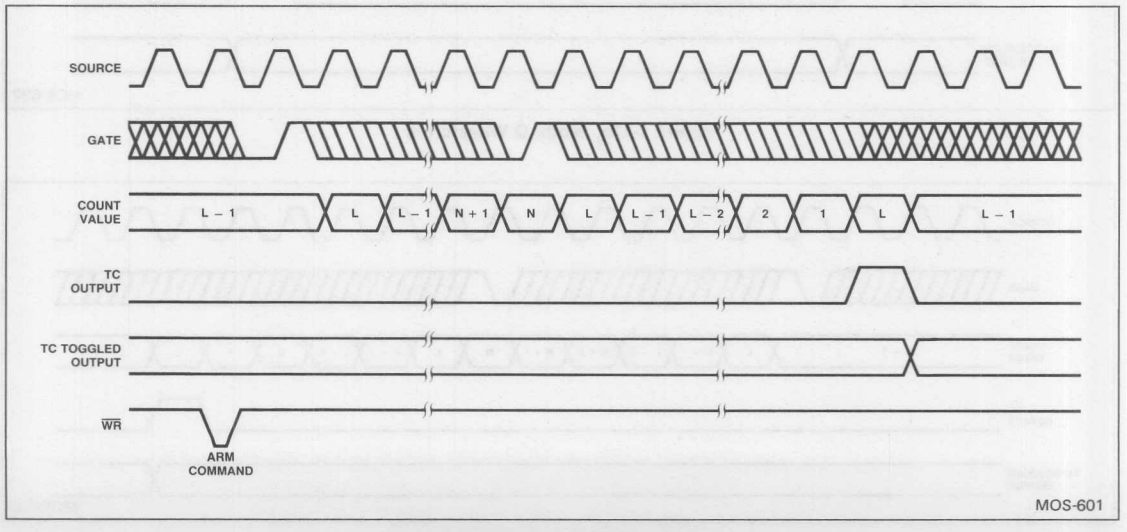


Figure 1-17o. Mode O Waveforms

MODE Q
Rate Generator with Synchronization
(Event Counter with Auto-Read/Reset)

CM15	CM14	CM13	CM12	CM11	CM10	CM9	CM8
LEVEL			X	X	X	X	X

CM7	CM6	CM5	CM4	CM3	CM2	CM1	CM0
1	0	1	X	X	X	X	X

Mode Q, shown in Figure 1-17q, provides a rate generator with synchronization or an event counter with auto-read/reset. The counter must first be issued an ARM command before counting can occur. Once armed, the counter will count all source edges which occur while the Gate is active and disregard those edges which occur while the Gate is inactive. This permits the Gate to turn the count process on and off. After the issuance of an ARM command and the application of an active Gate, the counter will count to TC repetitively. On each TC the counter will reload itself from the Load register. The counter may be retriggered at any time by presenting an active-going Gate edge to the Gate input. The retriggering Gate edge will transfer the contents of the counter into the Hold register. The first qualified source edge after the retriggering Gate edge will transfer the contents of the Load register into the Counter. Counting will resume on the second qualified source edge after the retriggering gate edge. Qualified source edges are active-going edges which occur while the Gate is active.

MODE R

Retriggerable One-Shot

CM15	CM14	CM13	CM12	CM11	CM10	CM9	CM8
EDGE			X	X	X	X	X

CM7	CM6	CM5	CM4	CM3	CM2	CM1	CM0
1	0	1	X	X	X	X	X

Mode R, shown in Figure 1-17r, is similar to Mode Q, except that edge gating rather than level gating is used. In other words, rather than use the Gate level to qualify which source edges to count, Gate edges are used to start the counting operation. The counter must be armed before application of the triggering Gate edge; Gate edges applied to a disarmed counter are disregarded. After application of a Gate edge, an armed counter will count all source edges until TC, irrespective of the Gate level. On the first TC the counter will be reloaded from the Load register and stopped. Subsequent counting will not occur until a new Gate edge is applied. All Gate edges applied to the counter, including the first used to trigger counting, initiate a retrigger operation. Upon application of a Gate edge, the counter contents are saved in the Hold register. On the first source edge after the retriggering Gate edge, the Load register contents will be transferred into the counter. Counting will resume on the second source edge after the retriggering Gate edge.

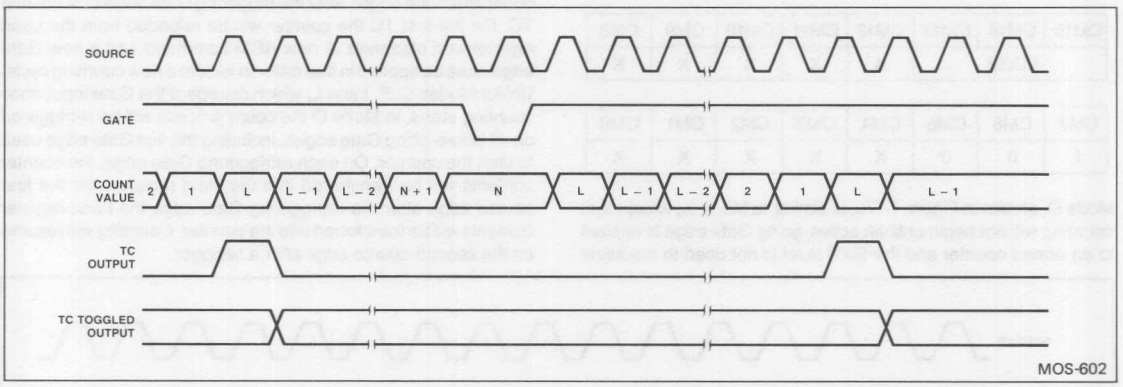


Figure 1-17q. Mode Q Waveforms

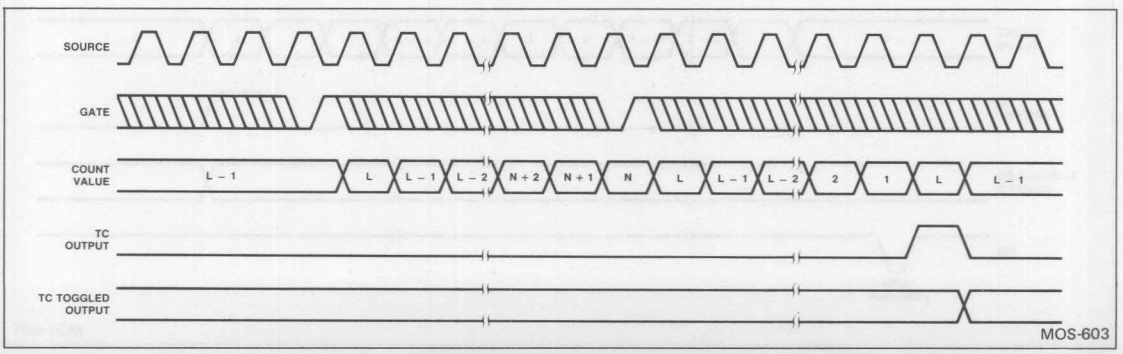


Figure 1-17r. Mode R Waveforms

MODE S

CM15	CM14	CM13	CM12	CM11	CM10	CM9	CM8
0	0	0	X	X	X	X	X

CM7	CM6	CM5	CM4	CM3	CM2	CM1	CM0
1	1	0	X	X	X	X	X

In this mode, the reload source for LOAD commands (irrespective of whether the counter is armed or disarmed) and for TC-initiated reloads is determined by the Gate input. The Gate input in Mode S is used only to select the reload source, not to start or modulate counting. When the Gate is Low, the Load register is used; when the Gate is High, the Hold register is used. Note the Low-Load, High-Hold mnemonic convention. Once armed, the counter will count to TC twice and then disarm itself. On each TC the counter will be reloaded from the reload source selected by the Gate. Following the second TC, an ARM command is required to start a new counting cycle. Mode S is shown in Figure 1-17s.

MODE V

Frequency-Shift Keying

CM15	CM14	CM13	CM12	CM11	CM10	CM9	CM8
0	0	0	X	X	X	X	X

CM7	CM6	CM5	CM4	CM3	CM2	CM1	CM0
1	1	1	X	X	X	X	X

Mode V, shown in Figure 1-17v, provides frequency-shift keying modulation capability. Gate operation in this mode is identical to that in Mode S. If the Gate is Low, a LOAD command or a TC-induced reload will reload the counter from the Load register. If the Gate is High, LOADs and reloads will occur from the Hold register. The polarity of the Gate only selects the reload source; it does not start or modulate counting. Once armed, the counter will count repetitively to TC. On each TC the counter will reload itself from the register determined by the polarity of the Gate. Counting will continue in this manner until a DISARM command is issued to the counter. Frequency shift keying may be obtained by specifying a TC Toggled output mode in the Counter Mode register. The switching of frequencies is achieved by modulating the Gate.

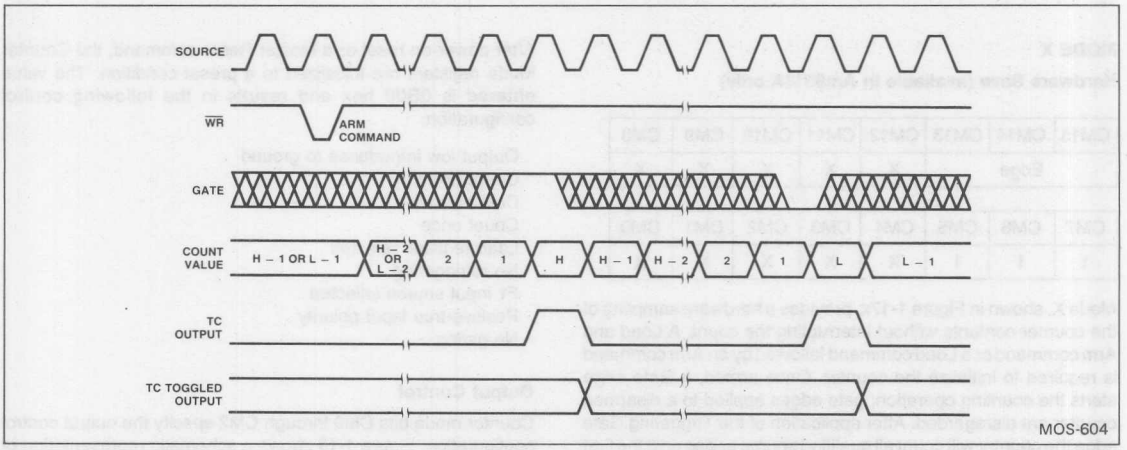


Figure 1-17s. Mode S Waveforms

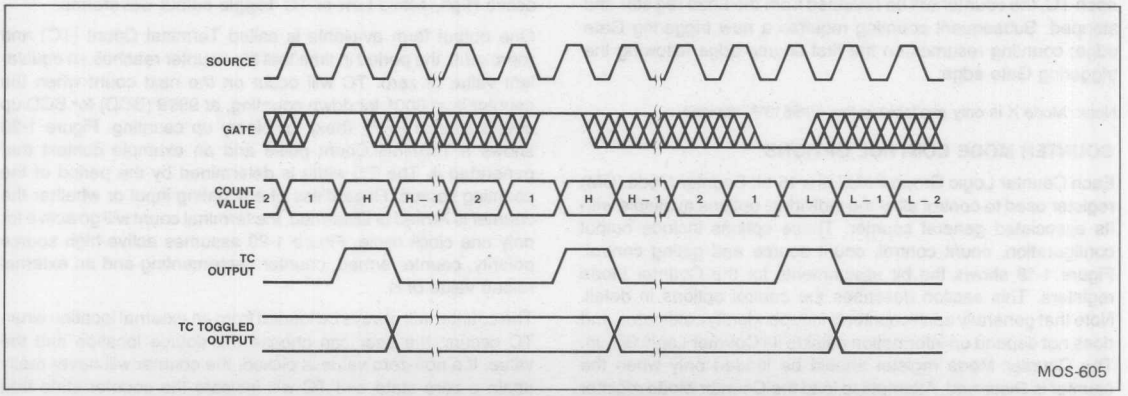


Figure 1-17v. Mode V Waveforms

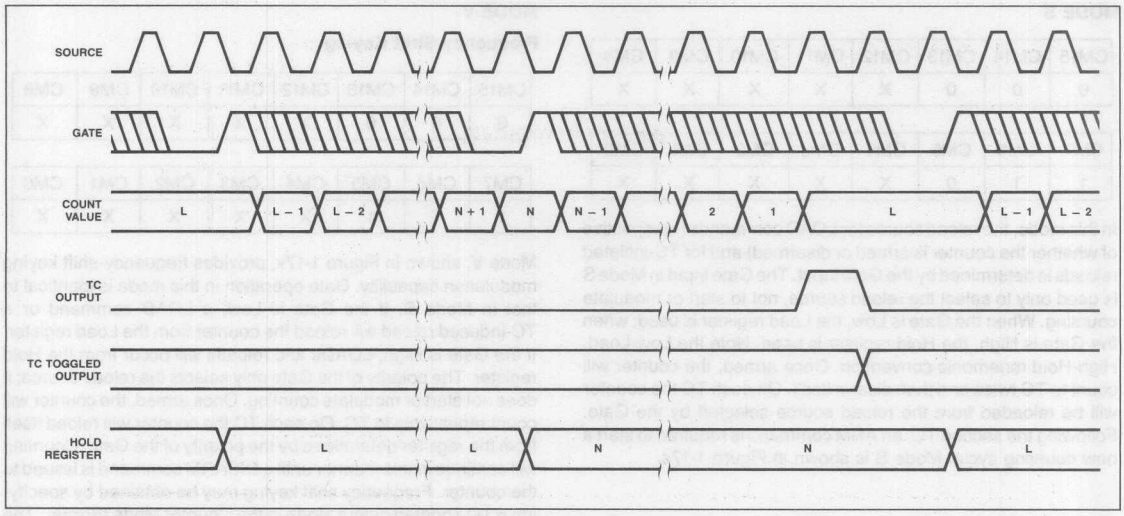


Figure 1-17x. Mode X Waveforms

MODE X

Hardware Save (available in Am9513A only)

CM15	CM14	CM13	CM12	CM11	CM10	CM9	CM8
Edge			X	X	X	X	X

CM7	CM6	CM5	CM4	CM3	CM2	CM1	CM0
1	1	1	X	X	X	X	X

Mode X, shown in Figure 1-17x, provides a hardware sampling of the counter contents without interrupting the count. A Load and Arm command or a Load command followed by an Arm command is required to initialize the counter. Once armed, a Gate edge starts the counting operation; gate edges applied to a disarmed counter are disregarded. After application of the Triggering Gate edge the counter will count all qualified source edges until the first TC, irrespective of the gate level. All gate edges applied during the counting sequence will store the current count in the Hold register, but they will not interrupt the counting sequence. On each TC, the counter will be reloaded from the Load register and stopped. Subsequent counting requires a new triggering Gate edge; counting resumes on the first source edge following the triggering Gate edge.

Note: Mode X is only available in the Am9513'A' devices.

COUNTER MODE CONTROL OPTIONS

Each Counter Logic Group includes a 16-bit Counter Mode (CM) register used to control all of the individual options available with its associated general counter. These options include output configuration, count control, count source and gating control. Figure 1-18 shows the bit assignments for the Counter Mode registers. This section describes the control options in detail. Note that generally each counter is independently configured and does not depend on information outside its Counter Logic Group. The Counter Mode register should be loaded only when the counter is Disarmed. Attempts to load the Counter Mode register when the counter is armed may result in erratic counter operation.

After power-on reset or a Master Reset command, the Counter Mode registers are initialized to a preset condition. The value entered is 0B00 hex and results in the following control configuration:

Output low impedance to ground
Count down
Count binary
Count once
Load register selected
No retriggering
F1 input source selected
Positive-true input polarity
No gating

Output Control

Counter mode bits CM0 through CM2 specify the output control configuration. Figure 1-19 shows a schematic representation of the output control logic. The OUT pin may be off (a high impedance state), or it may be inactive with a low impedance to ground. The three remaining valid combinations represent the active High, active Low or TC Toggle output waveforms.

One output form available is called Terminal Count (TC) and represents the period in time that the counter reaches an equivalent value of zero. TC will occur on the next count when the counter is at 0001 for down counting, at 9999 (BCD) for BCD up counting or at FFFF (hex) for binary up counting. Figure 1-20 shows a Terminal Count pulse and an example context that generated it. The TC width is determined by the period of the counting source. Regardless of any gating input or whether the counter is Armed or Disarmed, the terminal count will go active for only one clock cycle. Figure 1-20 assumes active-high source polarity, counter armed, counter decrementing and an external reload value of K.

The counter will always be loaded from an external location when TC occurs; the user can choose the source location and the value. If a non-zero value is picked, the counter will never really attain a zero state and TC will indicate the counter state that would have been zero had no parallel transfer occurred.

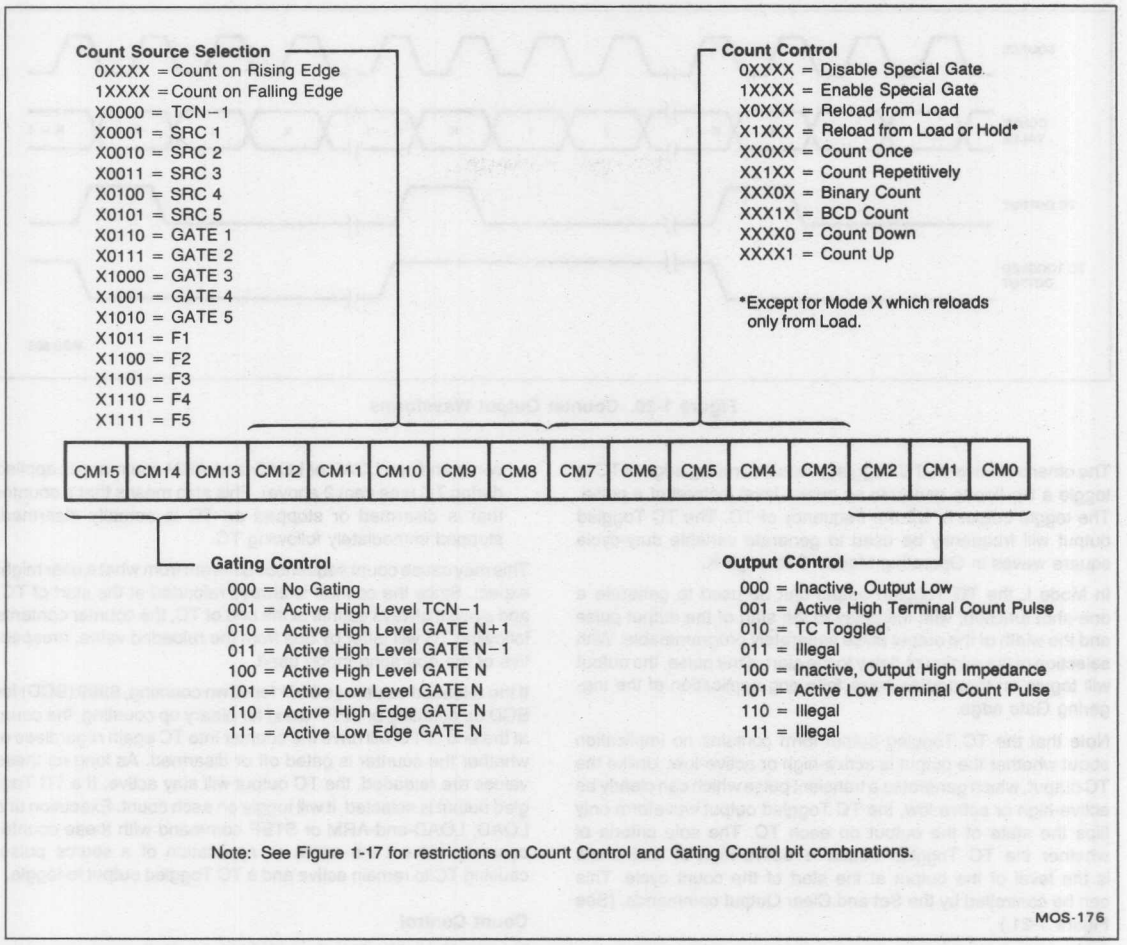


Figure 1-18. Counter Mode Register Bit Assignments

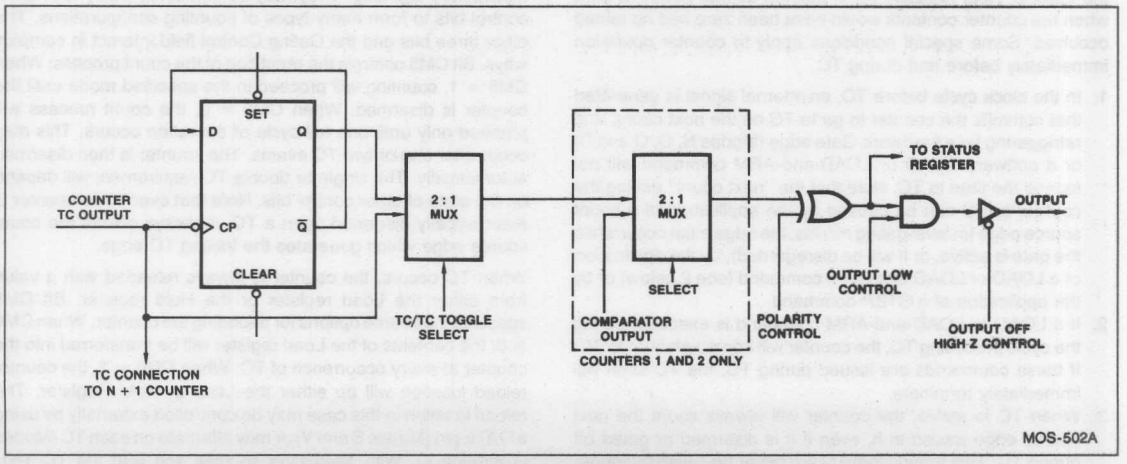


Figure 1-19. Output Control Logic

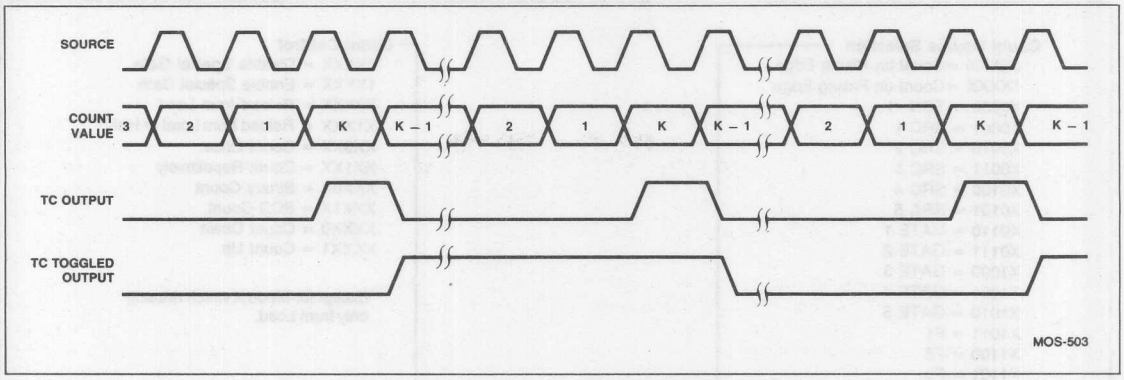


Figure 1-20. Counter Output Waveforms

The other output form, TC Toggled, uses the trailing edge of TC to toggle a flip-flop to generate an output level instead of a pulse. The toggle output is 1/2 the frequency of TC. The TC Toggled output will frequently be used to generate variable duty-cycle square waves in Operating Modes G through K.

In Mode L the TC Toggled output can be used to generate a one-shot function, with the delay to the start of the output pulse and the width of the output pulse separately programmable. With selection of the minimum delay to the start of the pulse, the output will toggle on the source pulse following application of the triggering Gate edge.

Note that the TC Toggled output form contains no implication about whether the output is active-high or active-low. Unlike the TC output, which generates a transient pulse which can clearly be active-high or active-low, the TC Toggled output waveform only flips the state of the output on each TC. The sole criteria of whether the TC Toggled output is active-high or active-low is the level of the output at the start of the count cycle. This can be controlled by the Set and Clear Output commands. (See Figure 1-21.)

TC (Terminal Count)

On each Terminal Count (TC), the counter will reload itself from the Load or Hold register. TC is defined as that period of time when the counter contents would have been zero had no reload occurred. Some special conditions apply to counter operation immediately before and during TC.

1. In the clock cycle before TC, an internal signal is generated that commits the counter to go to TC on the next count, and retriggering by a hardware Gate edge (Modes N, O, Q and R) or a software LOAD or LOAD-and-ARM command will not extend the time to TC. Note that the "next count" driving the counter to TC can be caused by the application of a count source edge (in level gating modes, the edge must occur while the gate is active, or it will be disregarded), by the application of a LOAD or LOAD-and-ARM command (see 2 below) or by the application of a STEP command.
2. If a LOAD or LOAD-and-ARM command is executed during the cycle preceding TC, the counter will immediately go to TC. If these commands are issued during TC, the TC state will immediately terminate.
3. When TC is active, the counter will always count the next source edge issued to it, even if it is disarmed or gated off during TC. This means that TC will never be active for longer than one count period and it may, in fact, be shorter if a STEP command or a LOAD or LOAD-and-ARM command is applied during TC (see item 2 above). This also means that a counter that is disarmed or stopped on TC is actually disarmed/stopped immediately following TC.

This may cause count sequences different from what a user might expect. Since the counter is always reloaded at the start of TC, and since it always counts at the end of TC, the counter contents following TC will differ by one from the reloaded value, irrespective of the operating mode used.

If the reloaded value was 0001 for down counting, 9999 (BCD) for BCD up counting or FFFF (hex) for binary up counting, the count at the end of TC will drive the counter into TC again regardless of whether the counter is gated off or disarmed. As long as these values are reloaded, the TC output will stay active. If a TC Toggled output is selected, it will toggle on each count. Execution of a LOAD, LOAD-and-ARM or STEP command with these counter contents will act the same as application of a source pulse, causing TC to remain active and a TC Toggled output to toggle.

Count Control

Counter Mode bits CM3 through CM7 specify the various options available for direct control of the counting process. CM3 and CM4 operate independently of the others and control up/down and BCD/binary counting. They may be combined freely with other control bits to form many types of counting configurations. The other three bits and the Gating Control field interact in complex ways. Bit CM5 controls the repetition of the count process. When CM5 = 1, counting will proceed in the specified mode until the counter is disarmed. When CM5 = 0, the count process will proceed only until one full cycle of operation occurs. This may occur after one or two TC events. The counter is then disarmed automatically. The single or double TC requirement will depend on the state of other control bits. Note that even if the counter is automatically disarmed upon a TC, it always counts the count source edge which generates the trailing TC edge.

When TC occurs, the counter is always reloaded with a value from either the Load register or the Hold register. Bit CM6 specifies the source options for reloading the counter. When CM6 = 0, the contents of the Load register will be transferred into the counter at every occurrence of TC. When CM6 = 1, the counter reload location will be either the Load or Hold Register. The reload location in this case may be controlled externally by using a GATE pin (Modes S and V) or may alternate on each TC (Modes G through L). With alternating sources and with the TC Toggled output selected, the duty cycle of the output waveform is

controlled by the relative Load and Hold values and very fine resolution of duty cycle ratios may be achieved.

Bit CM7 controls the special gating functions that allow retriggering and the selection of Load or Hold sources for counter reloading. The use and definition of CM7 will depend on the status of the Gating Control field and bits CM5 and CM6.

Hardware Retriggering

Whenever hardware retriggering is enabled (Modes N, O, Q and R) all active going Gate edges initiate retrigger operations. On application of the Gate edge, the counter contents will be transferred to the Hold register. On the first qualified source edge after application of the retriggering Gate edge, the Load register contents will be transferred into the counter. (Qualified source edges are edges which occur while the counter is gated on and Armed.)

This means that if level gating is used, the edge occurring on active-going gate transitions will initiate a retrigger. Similarly, when edge gating is enabled, an edge used to start the counter will also initiate a retrigger. The first count source edge applied after the Gate edge will not increment/decrement the counter but retrigger it.

If a Load, Load and Arm, or Step Command occur between the retriggering Gate edge and the first qualified source edge, it will be interpreted as a source edge and transfer the Load register contents into the counter. Thereafter, the counter will count all qualified source edges.

When some form of Gating is specified, CM7 controls hardware retriggering. In this case, when CM7 = 0 hardware retriggering does not occur; when CM7 = 1 the counter is retriggered any time an active-going Gate edge occurs. Retriggering causes the counter value to be saved in the Hold register and the Load register contents to be transferred into the counter.

When No Gating is specified, the definition of CM7 changes. In this case, when CM7 = 0 the Gate input has no effect on the counting; when CM7 = 1 the Gate input specifies the source (selecting either the Load or Hold register) used to reload the counter when TC occurs. Figure 1-16 shows the various available control combinations for these interrelated bits.

Count Source Selection

Counter Mode bits CM8 through CM12 specify the source used as input to the counter and the active edge that is counted. Bit CM12 controls the polarity for all the sources; logic zero counts rising edges and logic one counts falling edges. Bits CM8 through CM11 select 1 of 16 counting sources to route to the counter input. Five of the available inputs are internal frequencies derived from the internal oscillator (see Figure 1-15 for frequency assignments). Ten of the available inputs are interface pins; five are labeled SRC and five are labeled GATE.

The 16th available input is the TC output from the adjacent lower-numbered counter. (The Counter 5 TC wraps around to the Counter 1 input.) This option allows internal concatenating that permits very long counts to be accumulated. Since all five counters may be concatenated, it is possible to configure a counter that is 80-bits long on one Am9513 chip. When TCN−1 is the source, the count ripples between the connected counters. External connections can also be made, and can use the toggle bit for even longer counts. This is easily accomplished by selecting a TC Toggled output mode and wiring OUTN to one of the SRC inputs.

Gating Control

Counter Mode bits CM15, CM14, CM13 specify the hardware gating options. When "no gating" is selected (000) the counter will proceed unconditionally as long as it is armed. For any other gating mode, the count process is conditioned by the specified gating configuration.

For a code of 100 in this field, counting can proceed only when the pin labeled GATEN associated with Counter N is at a logic high level. When it goes low, counting is simply suspended until the Gate goes high again. A code of 101 performs the same function with an opposite active polarity. Codes 010 and 011 offer the same function as 100, but specify alternate input pins as Gating Sources. This allows any of three interface pins to be used as gates for a given counter. On Counter 4, for example, pin 34, pin 35 or pin 36 may be used to perform the gating function. This also allows a single Gate pin to simultaneously control up to three counters. Counters 1 and 5 are considered adjacent when using TCN − 1 (001), Gate N + 1 (010) and Gate N − 1 (011) controls.

For codes of 110 or 111 in this field, counting proceeds after the specified active Gate edge until one or two TC events occur. Within this interval the Gate input is ignored, except for the retriggering option. When repetition is selected, a cycle will be repeated as soon as another Gate edge occurs. With repetition selected, any Gate edge applied after TC goes active will start a new count cycle. Edge gating is useful when implementing a digital single-shot since the gate can serve as a convenient firing trigger.

A 001 code in this field selects the TC output from the adjacent lower-numbered counter as the gate. Thus, one counter may be configured to generate a counting "window" for another counter.

COMMAND DESCRIPTIONS

The command set for the Am9513 allows the host processor to customize and manage the operating modes and features for particular applications, to initialize and update both the internal data and control information, and to manipulate operating bits during operation. Commands are entered directly into the 8-bit Command register by writing into the Control port (see Figure 1-8).

All available commands are described in the following text. Figure 1-21 summarizes the command codes and includes a brief description of each function. Figure 1-22 shows all the unused code combinations; unused codes should not be entered into the Command register since undefined activities may occur.

Six of the command types are used for direct software control of the counting process and they each contain a 5-bit S field. In a linear-select fashion, each bit in the S field corresponds to one of the five general counters (S1 = Counter 1, S2 = Counter 2, etc.). When an S bit is a one, the specified operation is performed on the counter so designated; when an S bit is a zero, no operation occurs for the corresponding counter. This type of command format has three basic advantages. It saves host software by allowing any combination of counters to be acted on by a single command. It allows simultaneous action on multiple counters where synchronization of commands is important. It allows counter-specific service routines to control individual counters without needing to be aware of the operating context of other counters.

Three of the commands use a 3-bit binary code (N4, N2, N1) to identify the affected counter (a 001 programs counter 1, etc.). Unlike the previously mentioned commands, these commands allow you to program only one counter at a time.

Command Code								Command Description
C7	C6	C5	C4	C3	C2	C1	C0	
0	0	0	E2	E1	G4	G2	G1	Load Data Pointer register with contents of E and G fields. (G ≠ 000, G ≠ 110)
0	0	1	S5	S4	S3	S2	S1	Arm counting for all selected counters
0	1	0	S5	S4	S3	S2	S1	Load contents of specified source into all selected counters
0	1	1	S5	S4	S3	S2	S1	Load and Arm all selected counters
1	0	0	S5	S4	S3	S2	S1	Disarm and Save all selected counters
1	0	1	S5	S4	S3	S2	S1	Save all selected counters in Hold register
1	1	0	S5	S4	S3	S2	S1	Disarm all selected counters
1	1	1	0	1	N4	N2	N1	Set Toggle out (High) for counter N (001 ≤ N ≤ 101)
1	1	1	0	0	N4	N2	N1	Clear Toggle out (Low) for counter N (001 ≤ N ≤ 101)
1	1	1	1	0	N4	N2	N1	Step counter N (001 ≤ N ≤ 101)
1	1	1	0	1	0	0	0	Set MM14 (Disable Data Pointer Sequencing)
1	1	1	0	1	1	1	0	Set MM12 (Gate off FOUT)
1	1	1	0	1	1	1	1	Set MM13 (Enter 16-bit bus mode)
1	1	1	0	0	0	0	0	Clear MM14 (Enable Data Pointer Sequencing)
1	1	1	0	0	1	1	0	Clear MM12 (Gate on FOUT)
1	1	1	0	0	1	1	1	Clear MM13 (Enter 8-bit bus mode)
1	1	1	1	1	0	0	0	Enable Prefetch for Write operations (Am9513'A' only)
1	1	1	1	1	0	0	1	Disable Prefetch for Write operations (Am9513'A' only)
1	1	1	1	1	1	1	1	Master reset

Figure 1-21. Am9513 Command Summary

	C7	C6	C5	C4	C3	C2	C1	C0
	1	1	1	1	0	0	0	0
	1	1	1	1	0	1	1	0
	1	1	1	1	0	1	1	1
	0	0	0	X	X	1	1	0
	0	0	0	X	X	0	0	0
*	1	1	1	1	1	X	X	X

*Unused except when XXX = 111, 001 or 000.

Figure 1-22. Am9513 Unused Command Codes

Arm Counters

Coding:	C7	C6	C5	C4	C3	C2	C1	C0
	0	0	1	S5	S4	S3	S2	S1

Description: Any combination of counters, as specified by the S field, will be enabled for counting. A counter must be armed before counting can commence. Once armed, the counting process may be further enabled or disabled using the hardware gating facilities. This command can only arm or do nothing for a given counter; a zero in the S field does not disarm the counter.

ARM and DISARM commands can be used to gate counter operation on and off under software control. DISARM commands entered while a counter is in the TC state will not take effect until the counter leaves TC. This ensures that the counter never latches up in a TC state. (The counter may leave the TC state because of application of a count source edge; execution of a LOAD or LOAD-and-ARM command; or execution of a STEP command.)

In modes which alternate reload sources (Modes G-L), the ARMing operation is used as a reset for the logic which determines which reload source to use on the upcoming TC. Following each ARM or LOAD-and-ARM command, a counter in one of these modes will reload from the Hold register on the first TC and alternate reload sources thereafter (reload from the Load register on the second TC, the Hold register on the third, etc.).

Load Counters

Coding:	C7	C6	C5	C4	C3	C2	C1	C0
	0	1	0	S5	S4	S3	S2	S1

Description: Any combination of counters, as specified in the S field, will be loaded with previously entered values. The source of information for each counter will be either the associated Load register or the associated Hold register, as determined by the operating configuration in the Mode register. The Load/Hold contents are not changed. This command will cause a transfer independent of any current operating configuration for the counter. It will often be used as a software retrigger, or as counter initialization prior to active hardware gating.

If a LOAD or LOAD-and-ARM command is executed during the cycle preceding TC, the counter will go immediately to TC. This occurs because the LOAD operation is performed by generating a pseudo-count pulse, internal to the Am9513, and the Am9513 is expecting to go into TC on the next count pulse. The reload source used to reload the counter will be the same as that which would have been used if the TC were generated by a source edge rather than by the LOAD operation.

Execution of a LOAD or LOAD-and-ARM command while a counter is in TC will cause the TC to end. For Armed counters in

all modes except S or V, the LOAD source used will be that to be used for the upcoming TC. (The LOADing operation will not alter the selection of reload source for the upcoming TC.) For Disarmed counters in modes except S or V, the reload sources used will be the LOAD register. For modes S or V, the reload source will be selected by the GATE input, regardless of whether the counter is Armed or Disarmed.

Special considerations apply when modes with alternating reload sources are used (Modes G-L). If a LOAD command drives the counter to TC in these modes, the reload source for the next TC will be from the opposite reload location. In other words, the LOAD-generated TC will cause the reload sources to alternate just as a TC generated by a source edge would. Note that if a second LOAD command is issued during the LOAD-generated TC (or during any other TC, for that matter) the second LOAD command will terminate the TC and cause a reload from the source designated for use with the next TC. The second LOAD will not alter the reload source for the next TC since the second LOAD does not generate a TC; reload sources alternate on TCs only, not on LOAD commands.

Load and Arm Counters

Coding:

C7	C6	C5	C4	C3	C2	C1	C0
0	1	1	S5	S4	S3	S2	S1

Description: Any combination of counters, as specified in the S field, will be first loaded and then armed. This command is equivalent to issuing a LOAD command and then an ARM command.

A LOAD-and-ARM command which drives a counter to TC generates the same sequence of operations as execution of a LOAD command and then an ARM command. In modes which disarm on TC (Modes A-C and N-O, and Modes G-I and S if the current TC is the second in the cycle) the ARM part of the LOAD-and-ARM command will re-enable counting for another cycle. In modes which alternate reload sources (Modes G-L) the ARMing operating will cause the next TC to reload from the HOLD register, irrespective of which reload source the current TC used.

Disarm Counters

Coding:

C7	C6	C5	C4	C3	C2	C1	C0
1	1	0	S5	S4	S3	S2	S1

Description: Any combination of counters, as specified by the S field, will be disabled from counting. A disarmed counter will cease all counting independent of other control conditions. The only exception to this is that a counter in the TC state will always count once, in order to leave TC, before DISARMing. This count may be generated by a source edge, by a LOAD or LOAD-and-ARM command (the LOAD-and-ARM command will negate the DISARM command) or by a STEP command. A disarmed counter may be updated using the LOAD command and may be read using the SAVE command. A count process may be resumed using an ARM command. See the ARM command description for further details.

Save Counters

Coding:

C7	C6	C5	C4	C3	C2	C1	C0
1	0	1	S5	S4	S3	S2	S1

Description: Any combination of counters, as specified by the S field, will have their contents transferred into their associated Hold register. The transfer takes place without interfering with any counting that may be underway. This command will overwrite any previous Hold register contents. The SAVE command is designed to allow an accumulated count to be preserved so that it can be read by the host CPU at some later time.

Disarm and Save Counters

Coding:

C7	C6	C5	C4	C3	C2	C1	C0
1	0	0	S5	S4	S3	S2	S1

Description: Any combination of counters, as specified by the S field, will be disarmed and the contents of the counter will be transferred into the associated Hold registers. This command is identical to issuing a DISARM command followed by a SAVE command.

Set TC Toggle Output

Coding:

C7	C6	C5	C4	C3	C2	C1	C0
1	1	1	0	1	N4	N2	N1

$(001 \leq N \leq 101)$

Description: The initial output level for TC Toggle mode is set (High) for counter N selected by N4, N2, N1 = 001 (Counter 1) thru 101 (Counter 5) respectively. This command conditions the TC Toggle flip-flop (see Figure 1-19), but does not appear at the counter output unless TC Toggle mode (CM2, CM1, CM0 = 010) is selected.

Clear TC Toggle Output

Coding:

C7	C6	C5	C4	C3	C2	C1	C0
1	1	1	0	0	N4	N2	N1

$(001 \leq N \leq 101)$

Description: The initial output level for TC Toggle mode is Cleared (Low) for counter N selected by N4, N2, N1 = 001 (Counter 1) thru 101 (Counter 5) respectively. This command conditions the TC Toggle flip-flop (see Figure 1-19), but does not appear at the counter output unless TC Toggle mode (CM2, CM1, CM0 = 010) is selected.

Step Counter

Coding:

C7	C6	C5	C4	C3	C2	C1	C0
1	1	1	1	0	N4	N2	N1

$(001 \leq N \leq 101)$

Description: Counter N is incremented or decremented by one, depending on its operating configuration. If the Counter Mode register associated with the selected counter has its CM3 bit cleared to zero, this command will cause the counter to decrement by one. If CM3 is set to a logic high, this command will increment the counter by one. The STEP command will take effect even on a disarmed counter.

Load Data Pointer Register

Coding:

C7	C6	C5	C4	C3	C2	C1	C0
0	0	0	E2	E1	G4	G2	G1

$(G4, G2, G1 \neq 000, \neq 110)$

Description: Bits in the E and G fields will be transferred into the corresponding Element and Group fields of the Data Pointer register as shown in Figure 1-9. The Byte Pointer bit in the Data

Pointer register is set. Transfers into the Data Pointer only occur for G field values of 001, 010, 011, 100, 101 and 111. Values of 000 and 110 for G should not be used. See the "Setting the Data Pointer Register" section of this document for additional details.

Disable Data Pointer Sequencing

Coding:	C7	C6	C5	C4	C3	C2	C1	C0
	1	1	1	0	1	0	0	0

Description: This command sets Master Mode bit 14 without affecting other bits in the Master Mode register. MM14 controls the automatic sequencing of the Data Pointer register. Disabling the sequencing allows repetitive host processor access to a given internal location without repetitive updating of the Data Pointer. MM14 may also be controlled by loading a full word into the Master Mode register.

Enable Data Pointer Sequencing

Coding:	C7	C6	C5	C4	C3	C2	C1	C0
	1	1	1	0	0	0	0	0

Description: This command clears Master Mode bit 14 without affecting other bits in the Master Mode register. MM14 controls the automatic sequencing of the Data Pointer register. Enabling the sequencing allows sequential host processor access to several internal locations without repetitive updating of the Data Pointer. MM14 may also be controlled by loading a full word into the Master Mode register. See the "Data Pointer Register" section of this document for additional information on Data Pointer sequencing.

Enable 16-Bit Data Bus

Coding:	C7	C6	C5	C4	C3	C2	C1	C0
	1	1	1	0	1	1	1	1

Description: This command sets Master Mode bit 13 without affecting other bits in the Master Mode register. MM13 controls the multiplexer in the data bus buffer. When MM13 is set, no multiplexing takes place and all 16 external data bus lines are used to transfer information into and out of the STC. MM13 may also be controlled by loading the full Master Mode register in parallel.

Enable 8-Bit Data Bus

Coding:	C7	C6	C5	C4	C3	C2	C1	C0
	1	1	1	0	0	1	1	1

Description: This command clears Master Mode bit 13 without affecting other bits in the Master Mode register. MM13 controls the multiplexer in the data bus buffer. When MM13 is cleared, the multiplexer is enabled and 16-bit internal information is transferred eight bits at a time to the eight low-order external data bus lines. MM13 may also be controlled by loading the full Master Mode register in parallel.

Gate Off FOUT

Coding:	C7	C6	C5	C4	C3	C2	C1	C0
	1	1	1	0	1	1	1	0

Description: This command sets Master Mode bit 12 without affecting other bits in the Master Mode register. MM12 controls the output state of the FOUT signal. When gated off, the FOUT line will exhibit a low impedance to ground. MM12 may also be controlled by loading the full Master Mode register in parallel.

Gate On FOUT

Coding:	C7	C6	C5	C4	C3	C2	C1	C0
	1	1	1	0	0	1	1	0

Description: This command clears Master Mode bit 12 without affecting other bits in the Master Mode register. MM12 controls the output status of the FOUT signal. When MM12 is cleared, FOUT will become active and will drive out the selected and divided FOUT signal. MM12 may also be controlled by loading the full Master Mode register in parallel. When FOUT is gated on or off, a transient pulse may be generated on the FOUT signal.

Disable Prefetch for Write Operations

Coding:	C7	C6	C5	C4	C3	C2	C1	C0
	1	1	1	1	1	0	0	1

Description: This command disables the prefetch circuitry during Write operations (it does not affect Read operations). This reduces the write recovery time and allows the user to use block move instructions for initialization of the Am9513 registers. Once prefetch is disabled for writing, an Enable Prefetch for Write or a Reset command is necessary to re-enable the prefetch circuitry for writing. Note: This command is only available in Am9513'A' devices; it is an illegal command in the "non-A Am9513" device.

Enable Prefetch for Write Operations

Coding:	C7	C6	C5	C4	C3	C2	C1	C0
	1	1	1	1	1	0	0	0

Description: This command re-enables the prefetch circuitry for Write operations. It is used only to terminate the Disable Prefetch Command.

Master Reset

Coding:	C7	C6	C5	C4	C3	C2	C1	C0
	1	1	1	1	1	1	1	1

Description: The Master Reset command duplicates the action of the power-on reset circuitry. It disarms all counters, enters 0000 in the Master Mode, Load and Hold registers and enters 0B00 (hex) in the Counter Mode registers.

Following either a power-up or software reset, the LOAD command should be applied to all the counters to clear any that may be in a TC state. The Data Pointer register should also be set to a legal value, since reset does not initialize it. A complete reset operation is given in the following.

1. Using the procedure given in the "Command Initiation" section of this document, enter the FF (hex) command to perform a software reset.
2. Using the "Command Initiation" procedure, enter the LOAD command for all counters, opcode 5F (hex).
3. Using the procedure given in the "Setting the Data Pointer Register" section of this document, set the Data Pointer to a valid code. The legal Data Pointer codes are given in Figure 1-10.

The Master Mode, Counter Mode, Load and Hold registers can now be initialized to the desired values.

Chapter 2
Am9513 Interfacing

Chapter 2
Am9513 Interfacing

Am9513 – CPU INTERFACING

The Am9513 is designed to interface easily to both the Am8080A/8085A 8-bit family of CPUs and to the AmZ8000 16-bit family of CPUs. Master Mode register bit MM13 allows the user to program the Am9513 data bus for either an 8- or 16-bit width, allowing the Am9513's data bus to be tailored to match that of the host CPU.

Figure 2-1 shows an interface between the Am9513 and an Am8085A CPU. The Am9513 is configured to appear in the CPU's I/O space; connecting the IO/$\overline{M}$ output of the CPU to the $\overline{G2A}$ input of the decoder and tying G1 high will memory-map the Am9513. In the configuration shown, the Am9513 operates with an 8-bit data bus. Master Mode register bit MM13 should be 0 and data bus pins DB13-DB15 should be tied high as shown in the diagram.

Figure 2-2 shows a suggested connection diagram between the Am9513 and an AmZ8001* or AmZ8002* CPU. In this diagram the Am9513 appears in both Regular and Special I/O space, by virtue of the decoding of status lines ST1-ST3. Status line ST0 should be decoded also if it is necessary to separate the Regular and Special I/O spaces. The AmZ8136 is a latched decoder which stores the address information on the rising edge of $\overline{AS}$, providing the Am9513 with a stable $\overline{CS}$ for the duration of the transfer. The Am25LS158 multiplexer generates $\overline{RD}$ and $\overline{WR}$ from the CPU's $\overline{DS}$ and R/$\overline{W}$ lines. For maximum data bandwidth between the CPU and the Am9513, Master Mode register bit MM13 should be set to 1 to configure the Am9513 for a 16-bit data bus width. This can be accomplished by writing command opcode FFEF (hex) to the Am9513 following each reset and power-up.

CLOCK GENERATION

An internal oscillator is provided on the Am9513 for generation of timing frequencies to drive the source inputs for the five counters and the source for the FOUT pin. Note that a clock signal is not required for reads and writes to the Am9513. In applications which do not use the internal oscillator, the X2 input should be tied either High or Low to prevent accumulation of static charge. The X1 output is driven by an inverter contained in the Am9513 and accordingly, X1 should be left floating to avoid damaging the inverter's output stage.

Applications using the internal oscillator can drive the X1 and X2 inputs with an RC network, an external non-TTL level squarewave or a crystal. Figure 2-3 shows the recommended methods of connecting different frequency sources to the internal oscillator's input.

A crystal provides a highly accurate frequency source at moderate cost, and will usually be the preferred method of operation. The Am9513 is designed to use a crystal in parallel-resonant fundamental mode operation using the connection diagram shown in Figure 2-3a. Most series-resonant crystals can also be used, but the oscillator frequency will be different by up to a few percent from the series resonant crystal's rated frequency. Two ceramic capacitors should be connected between X1 and X2 to ground to ensure proper crystal loading. (The crystal loading is the capacitance the crystal should be driving to ensure on-frequency operation and reliable oscillator startup). Although the crystal sees the capacitors on X1 and X2 in series, and neglects the ground connection in the center, the use of two capacitors stabilizes the bias on the crystal by referencing it to ground and provides superior performance over the one capacitor equivalent circuit. Ceramic capacitors are the best type for this application because of their stability over time and temperature and their superior high frequency characteristics.

An RC network provides a very low cost frequency source but may exhibit large frequency variations over recommended power supply and temperature ranges, negating much of the precision available in the Am9513's counters. The RC connection is shown in Figure 2-3b. Note that although there is an internal resistor between X1 and X2, because this internal resistance is quite high, an external resistor should always be used in the RC operating configurations.

*Z8001 and Z8002 are trademarks of Zilog, Inc.

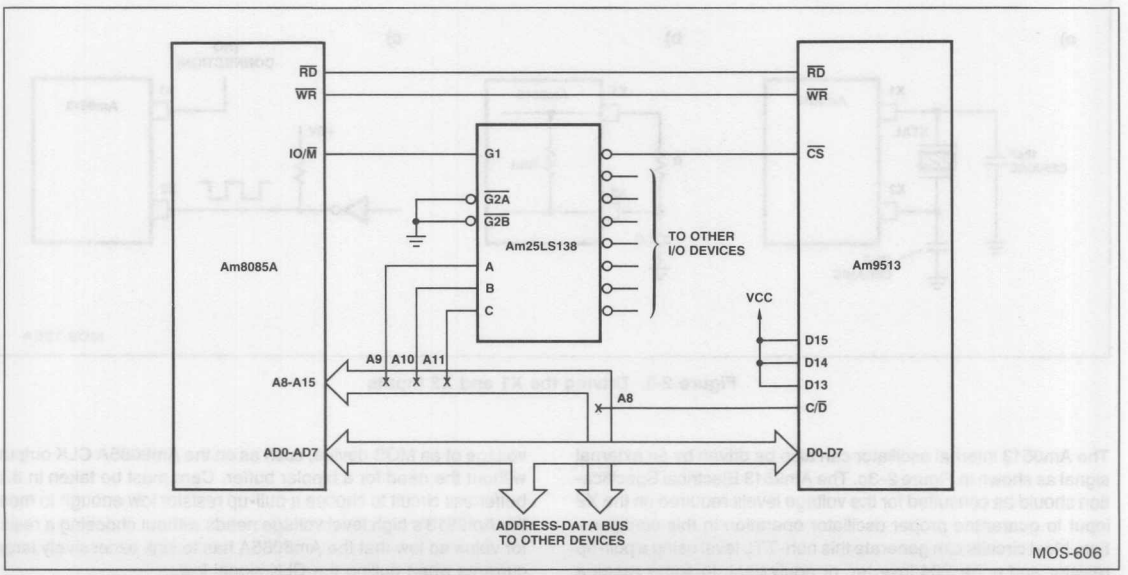


Figure 2-1. Am9513 – Am8085 Interfacing

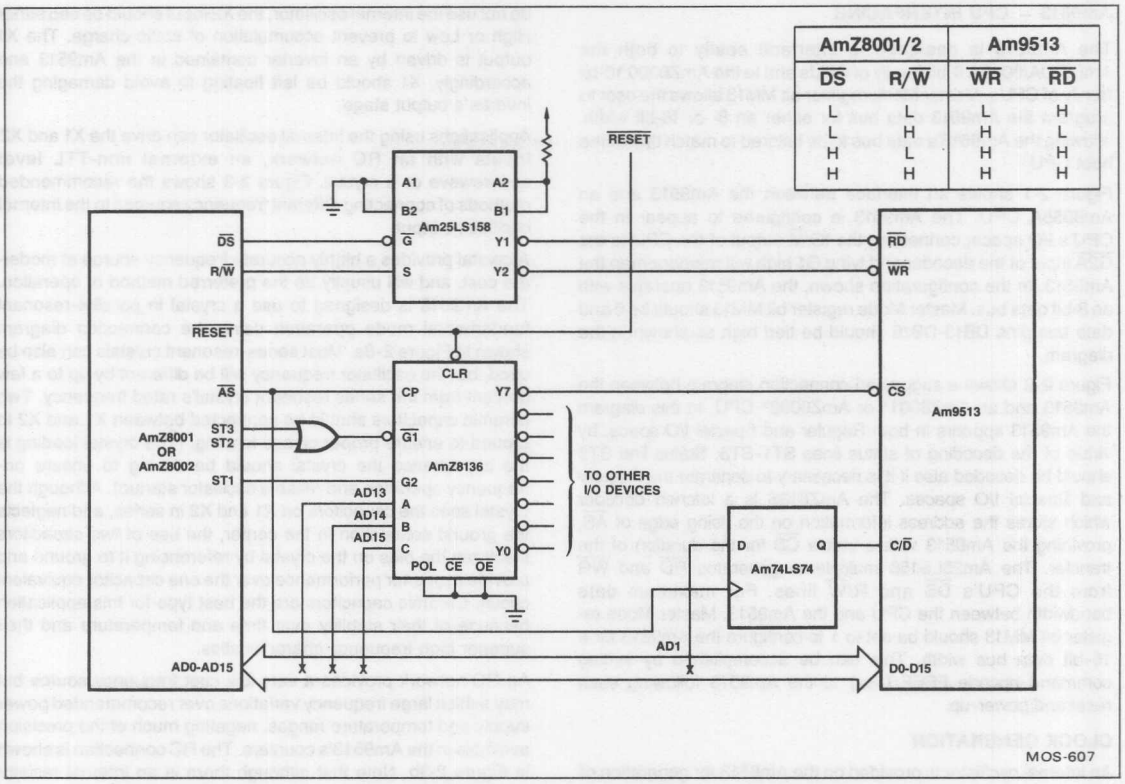


Figure 2-2. AmZ8001/8002 – Am9513 Interface

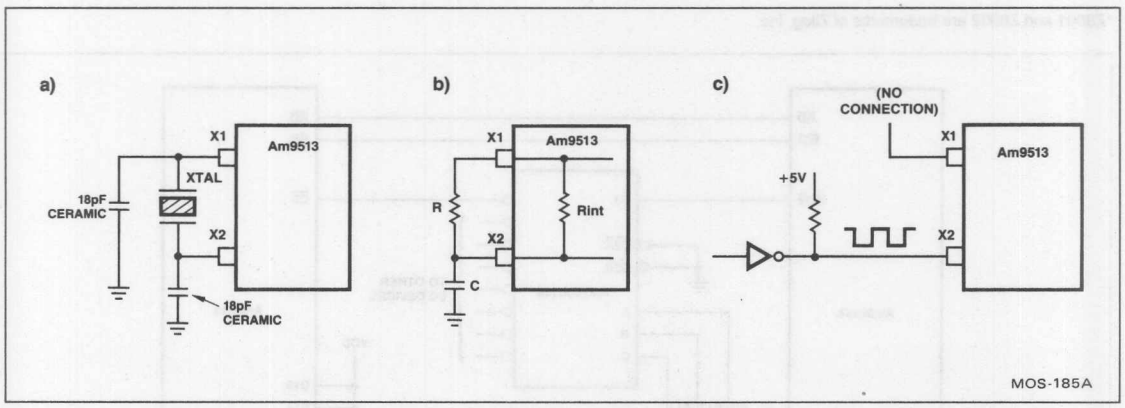


Figure 2-3. Driving the X1 and X2 Inputs

The Am9513 internal oscillator can also be driven by an external signal as shown in Figure 2-3c. The Am9513 Electrical Specification should be consulted for the voltage levels required on the X2 input to guarantee proper oscillator operation in this configuration. Most circuits can generate this non-TTL level using a pull-up resistor and a 74LS04 inverter, or equivalent. In some cases a pull-up resistor can be used to increase the high level output voltage of an MOS device, such as on the Am8085A CLK output, without the need for a bipolar buffer. Care must be taken in this bufferless circuit to choose a pull-up resistor low enough to meet the Am9513's high level voltage needs without choosing a resistor value so low that the Am8085A has to sink excessively large currents when pulling the CLK signal low.

REGISTER ACCESS

Information Transfer Protocols

The control signal configurations for all information transfers on the Am9513 data bus are summarized in Figure 2-4. The interface control logic assumes these conventions:

1. $\overline{RD}$ and $\overline{WR}$ are never active at the same time.
2. $\overline{RD}$, $\overline{WR}$ and C/$\overline{D}$ are ignored unless $\overline{CS}$ is Low.

The following discussion provides software oriented examples of Am9513 register accesses. Software examples are given for an Am8085 CPU with an 8-bit Am9513 data bus interface and for an AmZ8002 CPU with a 16-bit Am9513 data bus interface. The descriptions assume that the Am9513 Control port (CMDPRT) is located at address 12 (hex) and the Am9513 Data port (DATAPRT) is located at address 10 (hex). Later sections of this document present complete software listings for representative Am9513 applications.

Software Initialization

Figure 2-5 shows a Z8000 Software Initialization Sequence for the 9513. It is important to note the "DUMMY" LOAD COUNTER COMMAND; this insures proper operation of the part. The 16-bit mode command is not used for 8-bit CPUs. The sequence then is to Reset the device; Load all counters; Command 16-bit mode; set Data Pointer to the Master Mode Register; set Master Mode Register to desired value; set Data Pointer to counter #1 Mode Register and initialize counters to desired mode of operation. Note $\overline{CS}$ must be high during power-up or the internal reset circuitry will not function correctly. This will result in part ignoring all commands issued to it except software reset.

Command Initiation

Commands are issued to the Am9513 by writing the appropriate command code to the Am9513 Control Port. Figure 2-6 shows an example of command initiation, in this case opcode

Signal Configuration				Data Bus Operation
$\overline{CS}$	C/$\overline{D}$	$\overline{RD}$	$\overline{WR}$	
0	0	0	1	Transfer contents of register addressed by Data Pointer to the data bus.
0	0	1	0	Transfer contents of data bus to data register addressed by Data Pointer.
0	1	0	1	Transfer contents of Status register to data bus.
0	1	1	0	Transfer contents of data bus into Command register.
X	X	1	1	No transfer.
1	X	X	X	No transfer.
X	X	0	0	Illegal Condition.

Figure 2-4. Data Bus Transfers

AD (hex), which saves the contents of Counters 1, 3 and 4 in their associated Hold registers. In both the Am8080A/8085A and AmZ8002 coding examples, the command is loaded into an internal CPU register and output to the appropriate port. Note that in the AmZ8002 case since a 16-bit data bus interface is assumed the upper byte of data output to the Command port must be FF (hex).

The procedure for executing a command is as follows:

1. Establish the appropriate command on the DB0-DB7 lines. Figure 1-21 lists the command codes. When using the Am9513 in 16-bit mode, data bus lines DB8-DB15 should be set high during the write operation. In 8-bit data bus mode, DB13-DB15 should be set high during the write operation.

```
MACRO8000:          Version 2.0    9/19/80
MACZ  9513INIT S,P,L,O,W,D
INIT


0000                              %THIS IS A SAMPLE INITIALIZATION SEQUENCE
0000                              %FOR THE AM9513 COUNTER TIMER
0000
0000                        MODULE 'INIT';
0000
0000                        CONST   CMDPRT=12H,
0000                                DATAPRT=10H;
0000
0000   2101 FFFF            INIT:   LD      R1,#FFFF;
0004   3B16 0012                    OUT     CMDPRT,R1;      %SEND RESET
0008   2101 FF5F                    LD      R1,#FF5F;
000C   3B16 0012                    OUT     CMDPRT,R1;      %LOAD ALL COUNTERS
0010   2101 FFEF                    LD      R1,#FFEF;
0014   3B16 0012                    OUT     CMDPRT,R1;      %COMMAND 16 BIT MODE
0018   2101 FF17                    LD      R1,#FF17;
001C   3B16 0012                    OUT     CMDPRT,R1;      %POINT TO MASTER MODE REG
0020   2101 2CEF                    LD      R1,#2CEF;
0024   3B16 0010                    OUT     DATAPRT,R1;     %MASTER MODE SETTING
0028   2101 FF01                    LD      R1,#FF01;
002C   3B16 0012                    OUT     CMDPRT,R1;      %POINT TO CNTR 1 MODE REG
0030
0030                              END.
```

Figure 2-5. Am9513 Initialization

```
0100                    ORG     100H
                        ;
                        ;       AM9513 PORT ADDRESSES
                        ;
0012 =          CMDPRT  EQU     012H
0010 =          DATAPRT EQU     010H
                        ;
                        ;       AM9513 COMMAND INITIATION
                        ;
0100 3EAD               MVI     A,0ADH  ;SAVE CTRS. 1 3 & 4
0102 D312               OUT     CMDPRT
                        ;
                        ;
                        PAGE
```

a) 8080 Code

```
AM9513_EXAMPLES                 MACRO8000 AmZ8000 Assembler   1.0.1     Page 1


0000                            PROGRAM AM9513_EXAMPLES;
0000                            ORIGIN  0H;
0000                            %
0000                            %
0000                            %       AM9513 PORT ADDRESSES
0000                            %
0000                    CONST   CMDPRT=12H,
0000                            DATAPRT=10H;
0000                            %
0000                    AM9513_EXAMPLES:
0000                            %
0000                            %       AM9513 COMMAND INITIATION
0000                            %
0000    2102 FFAD               LD      R2,0FFADH;      %SAVE CTRS. 1 3 & 4
0004    3B26 0012               OUT     CMDPRT,R2;
0008                            %
0008                            %
0008                            EJECT;
```

b) AmZ8000 Code

Figure 2-6. Command Initiation Software

2. Establish a High on the C/$\overline{D}$ input.
3. Establish a Low on the $\overline{CS}$ input.
4. Establish a Low on the $\overline{WR}$ input.
5. Sometime after the minimum $\overline{WR}$ low pulse duration has been achieved, drive $\overline{WR}$ high, taking care the $\overline{CS}$, C/$\overline{D}$ and data setup times are met (see Timing Diagram).
6. After meeting the required $\overline{CS}$, C/$\overline{D}$ and data hold times, these signals can be changed (see Timing Diagram).

A new read or write operation to the Am9513 should not be performed until the write recovery time is met (see Timing Diagram in Electrical Specification.)

Setting the Data Pointer Register

The Data Pointer register selects which internal Am9513 register is to be accessed through the Data port. Setting the Data Pointer register automatically sets the Byte Pointer to 1, indicating a least significant byte is expected for 8-bit data bus interfacing. If Master Mode register bit MM14 = 0, the Data Pointer will automatically sequence through one of the cycles shown in Figure 1-11 after reading or writing each register, allowing sequential access to internal registers. If MM14 = 1, auto-sequencing is disabled and a single internal register can be repetitively accessed without reloading the Data Pointer. For convenience, bit MM14 can be set or cleared by software command.

The Pointer is set as follows:

1. Using Figures 1-9 and 1-10, select the appropriate Data Pointer Group and Element codes for the register to be accessed. Note that two codes are provided for the Hold registers, to accommodate both the Hold Cycle and Element Cycle autosequencing modes shown in Figure 1-11. If autosequencing is disabled, either Hold code may be used.
2. Using the "Writing to the Command Register" procedure given above, write the appropriate "Load Data Pointer" command to the Command register.

```
                        ;
                INTSR:  ;       INTERRUPT SERVICE ROUTINE
                        ;
                        ;       DISABLE INTERRUPTS
                        ;
0104 F3                 DI
                        ;
                        ;       SET DATA POINTER TO COUNTER 1 HOLD REG
                        ;
0105 3E19               MVI     A,019H
0107 D312               OUT     CMDPRT
                        ;
                        ;       ENABLE AUTO-SEQUENCING
                        ;
0109 3EE0               MVI     A,0E0H
010B D312               OUT     CMDPRT
                        ;
                        ;       CODE TO ACCESS REGISTERS
                        ;
                        ;       DISABLE AUTO-SEQUENCING
                        ;
010D 3EE8               MVI     A,0E8H
010F D312               OUT     CMDPRT
                        ;
                        ;       ENABLE INTERRUPTS AND RETURN
                        ;
0111 FB                 EI
0112 C9                 RET
                        ;
                        ;
                        PAGE
```

a) 8080 Code

```
AM9513_EXAMPLES                 MACRO8000 AmZ8000 Assembler   1.0.1     Page 2

0008                            %
0008                    INTSR:  %       INTERRUPT SERVICE ROUTINE
0008                            %
0008    7C00                    DI      NVI,VI;
000A                            %
000A                            %       SET DATA POINTER TO COUNTER 1 HOLD REG.
000A                            %
000A    2102 FF19               LD      R2,0FF19H;
000E    3B26 0012               OUT     CMDPRT,R2;
0012                            %
0012                            %       ENABLE AUTO-SEQUENCING
0012                            %
0012    2102 FFE0               LD      R2,0FFE0H;
0016    3B26 0012               OUT     CMDPRT,R2;
001A                            %
001A                            %       CODE TO ACCESS REGISTERS
001A                            %
001A                            %       DISABLE AUTO-SEQUENCING
001A                            %
001A    2102 FFE8               LD      R2,0FFE8H;
001E    3B26 0012               OUT     CMDPRT,R2;
0022                            %
0022                            %       ENABLE INTERRUPTS AND RETURN
0022                            %
0022    7C04                    EI      NVI,VI;
0024    7B00                    IRET;
0026                            %
0026                            %
0026                            EJECT;
```

b) AmZ8000 Code

Figure 2-7. Am9513 Interrupt Service Routine

In many systems the Am9513 counters will be serviced by interrupt routines. In such systems, it is important that the Am9513 service routines not be interrupted by another Am9513 service routine while register accesses are occurring. Consider, for example, an interrupt service routine which reads the Hold register value in the Counter 1 logic group. This routine will set the Data Pointer register and read the Hold register value. Consider the sequence of events which would occur if, after this routine set the Data Pointer registers, but before it read the Hold register, it was interrupted by a second Am9513 interrupt routine. This second routine might, for example, read the Counter 3 logic group Hold register value. When this second interrupt routine finishes, it returns control to the last half of the first interrupt routine. Because the second routine has changed the Data Pointer register, the first routine will not read the Hold register 1 contents. As can be seen from the above scenario, the sequence of operations of setting the Data Pointer register and accessing internal register locations must not be interrupted by another Am9513 service routine.

One way of ensuring that this restriction is met is to disable interrupts before setting the Data Pointer and not enabling interrupts until the register accesses are performed. Note that when auto-sequencing is used, interrupts should not be enabled until all registers have been accessed. An alternative method of meeting this restriction is to use software semaphores to prevent nesting of Am9513 service routines.

Figure 2-7 shows sample interrupt service routines which set the Data Pointer register to point to Counter 1's Hold register and enable Hold cycle auto-sequencing by clearing MM14. In the AmZ8002 case, a 16-bit data bus interface is assumed, requiring that the upper command byte be FF (hex). In the coding examples given interrupts are disabled and enabled by software command. Since the AmZ8002 architecture loads a new Flag and Control Word (FCW) when responding to an Interrupt request, the FCW loaded can disable further interrupts. This provides an alternative interrupt inhibiting mechanism for AmZ8002 systems and may be used in lieu of the software commands.

Reading the Status Register

The Am9513 Status register can be read either through the Control port or through the Data port. Figure 2-8 shows sample programs reading the Status register contents through the Control port into the accumulator (A register) of an Am8080A/8085A system or the R0 register of an AmZ8002 system. It is assumed that the AmZ8002 system has a 16-bit data bus; since the status register is only eight bits wide, the high byte of register R0 is undefined.

The procedure for reading the Status register through the Control port is given in the following.

1. Establish a High on the C/$\overline{D}$ input.
2. Establish a Low on the $\overline{CS}$ input.
3. After the appropriate $\overline{CS}$ and C/$\overline{D}$ setup time (see Timing Diagram) make $\overline{RD}$ Low.
4. Sometime after $\overline{RD}$ goes Low, the Status register contents will appear on the data bus. These lines will contain the information as long as $\overline{RD}$ is Low. If the state of an OUT pin changes while $\overline{RD}$ is Low, this will be reflected by a change in the information on the data bus.
5. $\overline{RD}$ can be driven High to conclude the read operation after meeting the minimum $\overline{RD}$ pulse duration.
6. $\overline{CS}$ and C/$\overline{D}$ can change after meeting the appropriate hold time requirements (see Timing Diagram).

A new read or write operation to the Am9513 should not be attempted until the read recovery time is met (see Timing Diagram in Electrical Specification).

```
                                ;
                                ;       CODE TO READ STATUS REGISTER
                                ;
        0113 DB12               IN      CMDPRT
                                ;
                                ;
                                PAGE
```

a) 8080 Code

```
AM9513_EXAMPLES                 MACRO8000 AmZ8000 Assembler   1.0.1     Page 3

0026                            %
0026                            %       CODE TO READ FROM STATUS REGISTER
0026                            %
0026    3B04 0012               IN      R0,CMDPRT;
002A                            %
002A                            %
002A                            EJECT;
```

b) AmZ8000 Code

Figure 2-8. Reading the Status Register

Reading From the Data Port

The registers which can be read from the Data port are the Load, Hold and Counter Mode registers for Counters 1 through 5, the Alarm registers for Counters 1 and 2, the Master Mode register and the Status register. The Status register can also be read from the Control port. Reading the Status register with a 16-bit data bus interface will return undefined information on DB8-DB15.

The procedure for reading these registers is as follows:

1. Prior to performing the actual read operation, the Data Pointer should be set to point to the register to be read, as outlined in the "Setting the Data Pointer" section of this document. In cases where auto-sequencing of the Data Pointer is used, the Pointer has to be set only once to the first register in the sequence. When auto-sequencing is disabled, repetitive accesses can be made to the same register without reloading the Data Pointer each time. Special care must be taken to reset the Data Pointer after issuing a command other than "Load Data Pointer" to the Am9513 or when operating a counter in modes N, O, Q or R. See the "Prefetch Circuit" section of this document for elaboration.
2. Establish a Low on the C/$\overline{D}$ input.
3. Establish a Low on the $\overline{CS}$ input.
4. Establish a Low on $\overline{RD}$ after waiting for the appropriate $\overline{CS}$ and C/$\overline{D}$ setup time (see Timing Diagram).
5. Sometime after $\overline{RD}$ goes Low, the register contents will appear on the data bus. In both 8- and 16-bit bus modes the low register byte will appear on DB0-DB7. In addition, in 16-bit bus mode, the upper register byte will appear on the DB8-DB15. For 8-bit bus mode, pins DB8-DB15 are not driven by the Am9513. This information will remain stable as long as $\overline{RD}$ is Low. If the register value is changed during the read, the change will not be reflected by a change in the data being read, for the reasons outlined in the "Prefetch Circuit" section of this document.
6. $\overline{RD}$ can be driven High to conclude the read operation after meeting the minimum $\overline{RD}$ pulse duration.
7. $\overline{CS}$ and C/$\overline{D}$ can change after meeting appropriate hold time requirements (see Timing Diagram).
8. After waiting the minimum read recovery time (see Timing Diagram), a new read or write operation can be started. For 8-bit bus mode, steps 2 through 7 should be repeated to read out the high register byte on DB0-DB7. (If the Status register is being read in 8-bit mode, the two reads will return the Status register each time. In 16-bit mode, reads from the Status register return undefined data on DB8-DB15.) The user is not required to drive $\overline{CS}$ or C/$\overline{D}$ High between successive reads or writes, although this is permissible.

As described in the "Setting the Data Pointer" register section, the Am9513 service routines should disable interrupts during Data port register accesses if the service routine could be interrupted by another service routine requiring access to Data port registers.

Figure 2-9 shows sample programs for reading a Data port register. The Am8080A/8085A code reads the data in two byte reads (low byte first) and assembles it into the HL register pair. The AmZ8002 program assumes that a 16-bit data interface is being used and reads the data into register R0 in a single word read. This code can be substituted into the sample interrupt service routines in Figure 2-7 in the place marked "Code to Access Registers."

```
                                ;
                                ;       CODE TO READ FROM DATA PORT REG.
                                ;
0115 DB10                       IN      DATAPRT
0117 6F                         MOV     L,A
0118 DB10                       IN      DATAPRT
011A 67                         MOV     H,A
                                ;
                                ;
                                PAGE
```

a) 8080 Code

```
AM9513_EXAMPLES                 MACRO8000 AmZ8000 Assembler   1.0.1     Page 4

002A                            %
002A                            %       CODE TO READ FROM DATA PORT REG.
002A                            %
002A    3B24 0010               IN      R2,DATAPRT;
002E                            %
002E                            %
002E                            EJECT;
```

b) AmZ8000 Code

Figure 2-9. Reading Through the Data Port

Writing to the Data Port

The registers which can be written to through the Data port are the Load, Hold and Counter Mode registers for Counters 1 through 5, the Alarm registers for Counters 1 and 2 and the Master Mode register. The procedure for writing to these registers is as follows:

1. Prior to performing the actual write operation, the Data Pointer should be set to point to the register to be written to, as outlined above in the "Setting the Data Pointer" section of this document. In cases where auto-sequencing of the Data Pointer is used, the Pointer has to be set only once to the first register in the sequence. When auto-sequencing is disabled, repetitive accesses can be made to the same register without reloading the Data pointer each time.
2. Establish the appropriate data on the DB0-DB7 lines (8-bit bus mode) or DB0-DB15 (16-bit bus mode). When using the 8-bit bus mode, data bus lines DB13-DB15 should be set High during the write operation and DB0-DB7 should be set to the lower data byte for the first write and to the upper data byte for the second write.
3. Establish a Low on the C/$\overline{D}$ input.
4. Establish a Low on the $\overline{CS}$ input.
5. Establish a Low on the $\overline{WR}$ input.
6. Drive $\overline{WR}$ High sometime after the minimum $\overline{WR}$ low pulse duration has been achieved, taking care the $\overline{CS}$, C/$\overline{D}$ and data setup times are met (see Timing Diagram).
7. After meeting the required $\overline{CS}$, C/$\overline{D}$ and data hold times, these signals can be changed (see Timing Diagram).
8. After meeting the write recovery time (see Timing Diagram) a new read or write operation can be performed. For the 8-bit bus mode, steps 2 through 7 should be repeated, this time placing the high data byte on pins DB0-DB7. The user is not required to drive $\overline{CS}$ or C/$\overline{D}$ High between successive reads or writes, although this is permissible.

As described in the "Setting the Data Pointer" section, Am9513 service routines should disable interrupts during Data port register accesses if the service routine could be interrupted by another service routine requiring access to the Data port registers.

Figure 2-10 shows sample programs for writing a 16-bit value to a Data port register. The Am8080A/8085A code loads the register by making two byte transfers (low byte first) to the Am9513 Data port. A 16-bit data bus interface is assumed for the AmZ8002 coding example; accordingly, a single word transfer can be used to load a register. This code can be substituted into the sample interrupt service routines in Figure 2-7 in the place marked "Code to Access Registers."

```
                        ;
                        ;       CODE TO WRITE TO DATA PORT REG.
                        ;
011B 7D                 MOV     A,L
011C D310               OUT     DATAPRT
011E 7C                 MOV     A,H
011F D310               OUT     DATAPRT
                        ;
                        ;
                        ;
0121                    END

A>
```

a) 8080 Code

```
AM9513_EXAMPLES                 MACRO8000 AmZ8000 Assembler   1.0.1     Page 5

002E                            %
002E                            %       CODE TO WRITE TO DATA PRT REG.
002E                            %
002E   3B26 0010                OUT     DATAPRT,R2;
0032                            %
0032                            %
0032                            %
0032                            END.
```

b) AmZ8000 Code

Figure 2-10. Writing Through the Data Port

Chapter 3
Concatenating Counters

Chapter 3
Concatenating Counters

CONCATENATING COUNTERS

The Am9513 counters may be concatenated in a number of different ways. These may be conceptually broken down into count up and count down concatenation. Count up concatenation will typically be used to count events with a precision greater than 16 bits. Count down concatenation is typically used to generate output frequencies of high resolution.

To simplify concatenation, the Am9513 provides an internal TC signal from the low order counter which can be selected as a count source in the high order counter's Counter Mode register. Thus, although any two counters can be concatenated with external strapping, usually adjacent counters will be used to allow use of this internal TC signal.

In count up concatenation, both the high and low order counter's Load register should be cleared to 0. The low order counter will start counting up from 0 and increment through 9999. (BCD counting is assumed throughout this discussion, although binary counting may, of course, be used). On the next source edge the low order counter will go to TC and reload 0 from the Load register. The active-going TC edge will also increment the high order counter. The counters continue counting in this manner with the high order counter incrementing each time the low order counter reaches TC. In the examples which follow, Counters 1 and 2 will be used as the low order and high order counters respectively.

In the first up concatenation configuration, shown in Figure 3-1, the counters do not use external gating and therefore will free run. The high order counter should use the TC output of the low order counter as a source. The high order counter should count on rising source edges and should be programmed for "no gating." The above requirements can be met by specifying 00 (hex) in the upper byte of the high order counter's Mode register. The low order counter should be programmed to count repetitively. The required Mode register settings for Counters 1 and 2 are shown in the figure; "don't care" bits are marked "X." Note that if the internal TC signal is used to concatenate to the upper counter, no restrictions are placed on the programming of the low order counter's Output Control field. Conversely, if external strapping is used to concatenate the counter, the low order counter should have an "Active High TC" output mode selected. Up count concatenation may also be used with either level or edge gating. For level gating, the count source may either be externally gated with external logic, or the low order counter may be programmed for level gating, as shown in Figure 3-2. In either case, the high order counter should be programmed for "no gating." Recall that while in the TC state, the counters will count all source pulses issued to them, irrespective of their gating or arming status. This can introduce counting errors when level gating is used in up count concatenation. If the gate goes inactive while the low order counter is in TC, the low order counter will count the next source edge, which drives it out of TC. The counter will then stop counting until the gate goes active again. This effectively introduces a 1 count error into the accumulated count. The maximum error that can be introduced is one extra count each time the gate is applied. This worst case error will occur only if the gate is always applied when the low order counter is in the TC state. For many applications which use the gate infrequently, this small potential error is of no significance. Applications sensitive to small count errors or applications with many gate-on, gate-off cycles should use external gating logic to inhibit source pulses.

Edge gating functions can also be used in up count concatenation. An edge gating circuit with concatenated counters should function in a logically identical manner to a single edge-gated counter. In other words, after an edge is applied to the concatenated counters, they should count until both reach TC. A new edge should be required to repeat the cycle. Direct concatenation of two counters as was done for level gating up count

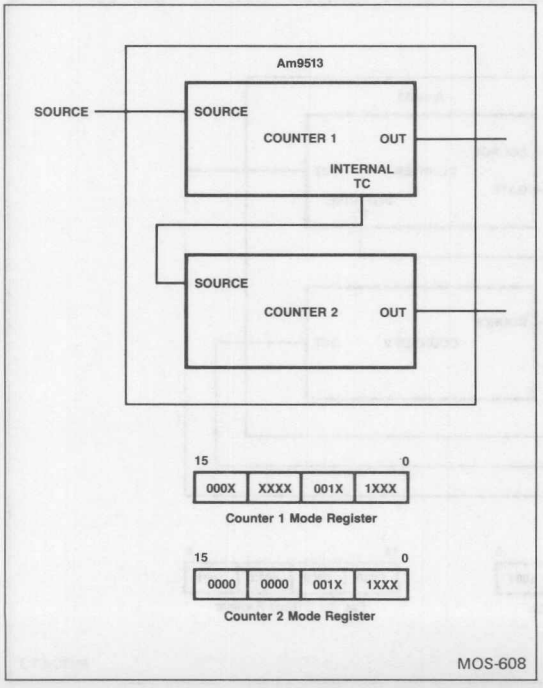


Figure 3-1. Count Up Concatenation with No Gating

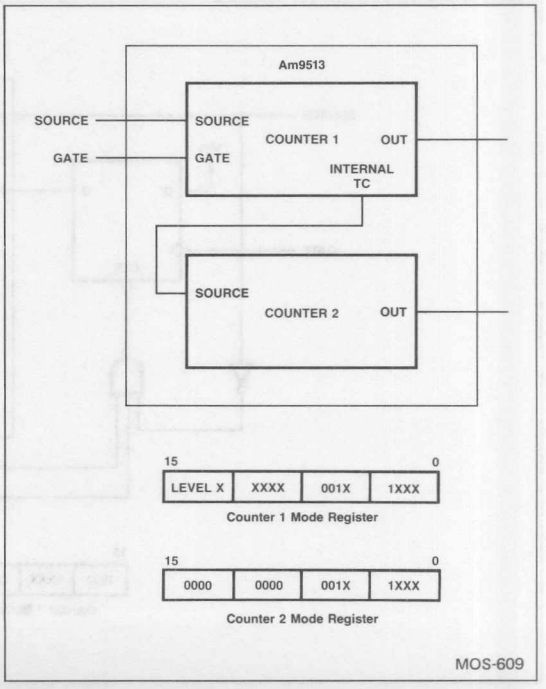


Figure 3-2. Count Up Concatenation with Level Gating

concatenation will not work. In such an arrangement, the low order counter, once triggered, will count to TC once and then stop, awaiting a new gate edge. This is unsatisfactory since we want the low order counter to continue counting until the high order counter reaches TC.

Figure 3-3 shows one method of concatenating counters for edge-triggered up counting. This method operates the counters in a similar arrangement to that used for level gating, with the requirement that each counter's output be programmed for an active-high TC pulse.

The external flip-flop is set by an external, synchronous gate signal. When both counters reach TC, the flip-flop is cleared. One potential problem exists with this scheme. Once the flip-flop clears, it will inhibit the low order counter's gate input. The low order counter will nevertheless count the next source edge, driving itself out of TC. However, the high order counter will remain in TC. When the next triggering gate edge is applied, the flip-flop will set, allowing counting to begin. When the low counter reaches its first TC, the rising TC edge will cause the high counter to leave TC. For a short period of time (the propagation delay of the high order counter from source to output), both TCs will again be active. This could potentially clear the flip-flop prematurely. To inhibit this, the source signal is added as an additional input to the NAND gate. If a relatively slow source is used with a high time greater than the total propagation delay from the source input of low order counter to the output of the high order counter, the source input on the NAND gate will inhibit clearing of the flip-flop during this transient.

The concatenation examples so far have assured that the counters are to "count repetitively," in the sense of counter mode register bit CM5. If "count once" operation is desired, in which the counters require an Arm command after each count cycle, different circuits are required.

When "no gating," "count once" operation is desired, the circuit in Figure 3-4 can be used. In this application, Counter 1 should be programmed for active-high level gating and Counter 2 should be programmed for a TC Toggled output. During counter initialization, the following set of commands should be used:

Initialize Counters' 1 and 2 Mode and Load registers
LOAD Counters 1 and 2
Clear Counter 2 output
ARM Counters 1 and 2.

The counters are now ready to count, but since Counter 2's output is low, Counter 1's gate will inhibit counting. To start counter operation, use the "Set Counter 2's output" command. The counters will then count applied source pulses until Counter 2 reaches TC and toggles its output, inhibiting Counter 1's gate. It can be seen that in this application, the "Set Counter 2's output" behaves as an ARM command. It is important that the counting rate be low enough to ensure that Counter 1's gate will not go inactive in close proximity to a source edge. High speed applications using a Counter 1 source period less than the propagation delay from Counter 1's source to Counter 2's output should use a flip-flop to synchronize Counter 2's output to Counter 1's source in order to meet timing parameters TGVEH and TEHGV in the Am9513 data sheet. High-speed applications will end the count cycle with a value slightly larger than 1 in Counter 1.

To add level gating to this "count once" feature simply involves the addition of an AND function before Counter 1's gate input. Now Counter 1 will be inhibited whenever Counter 2 toggles its output or whenever the external gate is driven low. Note that this circuit assumes the externally applied gate is synchronous to the count source; asynchronous gating signals should be synchronized with a flip-flop.

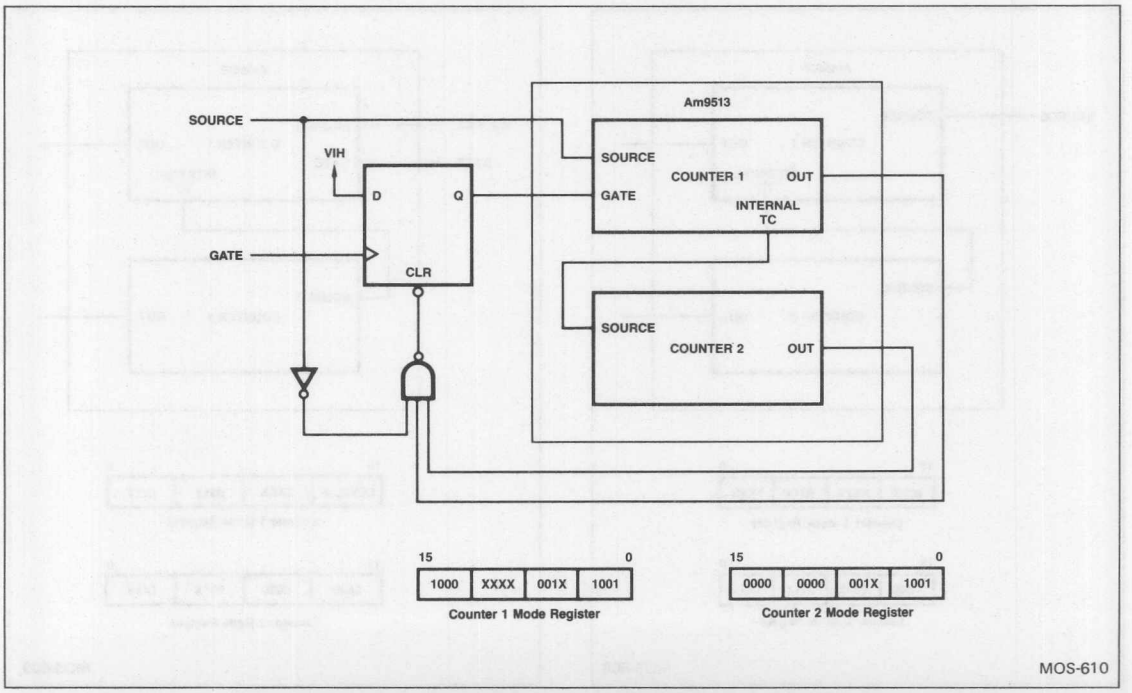


Figure 3-3. Count Up Concatenation with Edge Gating

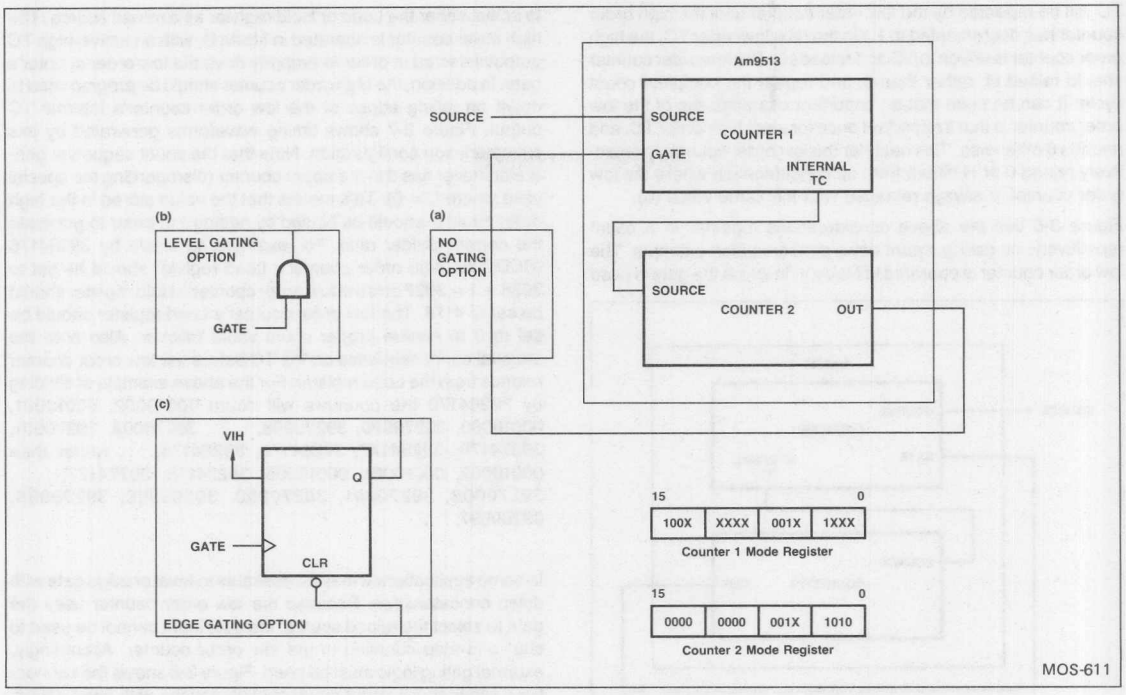


Figure 3-4. Count Up Concatenation with Count Once Feature

The final case of concatenated up counting comprises edge gating with the "count once" feature. This is achieved through a simple variation of the level gating configuration. An external gate signal sets the flip-flop and enables counting providing Counter 2's output is set. When Counter 2 reaches TC, its output will toggle (i.e., clear) and the flip-flop will clear, inhibiting further counting. To restart the counter in this configuration, the "Set Counter 2 output" command should be issued and a new gate edge should be applied in the order. As in the previous cases, the applied gate edge should be synchronous to the Counter 1 source.

In order to analyze down concatenation, it is useful to separately analyze the sequences followed for the high order and low order counters. Figure 3-5 shows a typical count down concatenation sequence, with the high order and low order count sequences labelled. The high order counter simply decrements from some initial value L until TC is reached. (In the following discussion and figures, L and H are used to represent the Load and Hold register contents respectively; K and N are used to represent arbitrary count values.) It is then reloaded with L and repeats the sequence. Note that the high order counter, in general, will never count to 0, since TC is generated by the source edge occurring while the counter contains 1 and TC reloads the initial value L. The count sequence is thus L, (L−1), . . . , 2, 1, L, (L−1), (L−2), . . . , 2, 1, L. The low order counter starts from some initial value H and counts down to TC. This TC output will be used to decrement the high order counter by 1. The low order counter is now reloaded with 0 and counts down through (assuming BCD counting) 9999 to 1. This sequence of reloading 0 and counting down to the next

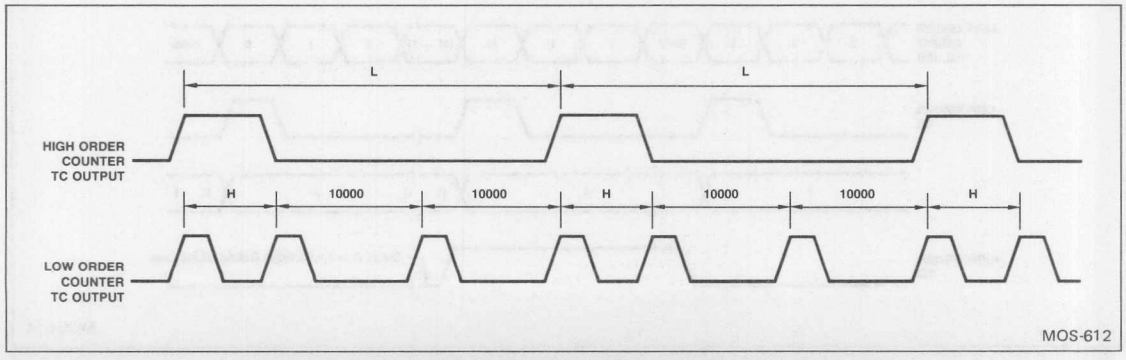


Figure 3-5. Conceptual Sequence for Count Down Concatenation

TC will be repeated by the low order counter until the high order counter has decremented to 1. On the next low order TC, the high order counter is driven to TC and reloads L. The low order counter should reload H, rather than 0, and repeat the complete count cycle. It can be seen that an important characteristic of the low order counter is that it reloads H once for each high order TC, and reloads 0 otherwise. This need for the low order counter to selectively reload 0 or H differs from up concatenation where the low order counter is always reloaded with the same value (0).

Figure 3-6 ties the above considerations together in a count repetitively, no gating, count down concatenation example. The low order counter is operated in Mode V, in which the gate is used to select either the Load or Hold register as a reload source. The high order counter is operated in Mode D, with an active-high TC output selected in order to properly drive the low order counter's gate. In addition, the high order counter should be programmed to count on falling edges of the low order counter's internal TC output. Figure 3-7 shows timing waveforms generated by this concatenation configuration. Note that the count sequence generated never has 0 in the upper counter (disregarding the special case where L = 0). This means that the value stored in the high order counter should be biased by adding 1 in order to generate the correct divider ratio. For example, to divide by 39264178 (BCD), the high order counter's Load register should be set to 3926 + 1 = 3927 and the low order counter's Hold register should be set to 4178. The low order counter's Load register should be set to 0 to ensure proper count value rollover. Also note the unusual count sequence on the TC before the low order counter reloads from the Load register. For the above example of dividing by 39264178 the counters will count 00010002, 00010001, 00010000, 39279999, 39279998, . . . , 39270002, 39270001, 39274178, 39264177, 39264176, 39264175, . . . rather than 00010002, 00010001, 00010000, 39274178, 39274177, . . . , 39270002, 39270001, 39270000, 39269999, 39269998, 39269997, ...

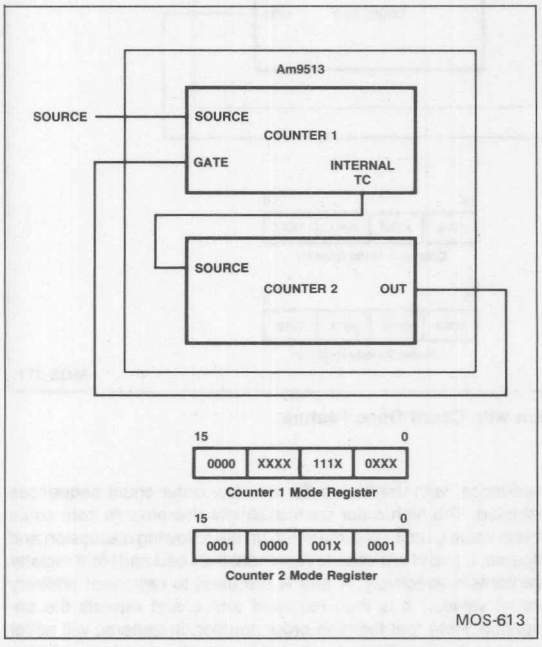


Figure 3-6. Count Down Concatenation

In some applications it may be desirable to level or edge gate with down concatenation. Because the low order counter uses the gate to select the reload source, the gate input cannot be used to start and stop counting in the low order counter. Accordingly, external gating logic must be used. Figure 3-8 shows the connections required for count down concatenations with level gating. Level gating is achieved by inhibiting source pulses when the gate goes inactive.

Edge gating, shown in Figure 3-9, uses an external gate signal to set an enabling flip-flop. The enabling flip-flop is cleared when both counters reach TC. The delay flip-flop ensures that one additional count occurs after both counters reach TC in order to drive the low order counter out of TC, thereby deactivating the enabling flip-flop's clear input. Note that the counters stop at an unusual point in the count sequence, ((L−1), (H−1) in Figure 3-7 or 3926 4177 for the earlier example) but this is not important

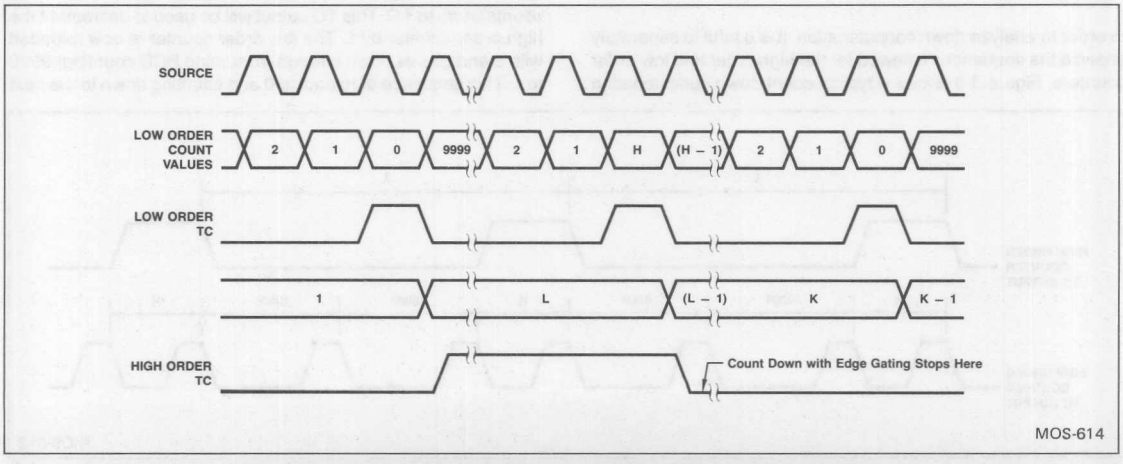


Figure 3-7. Timing Waveforms for Am9513 Count Down Concatenation

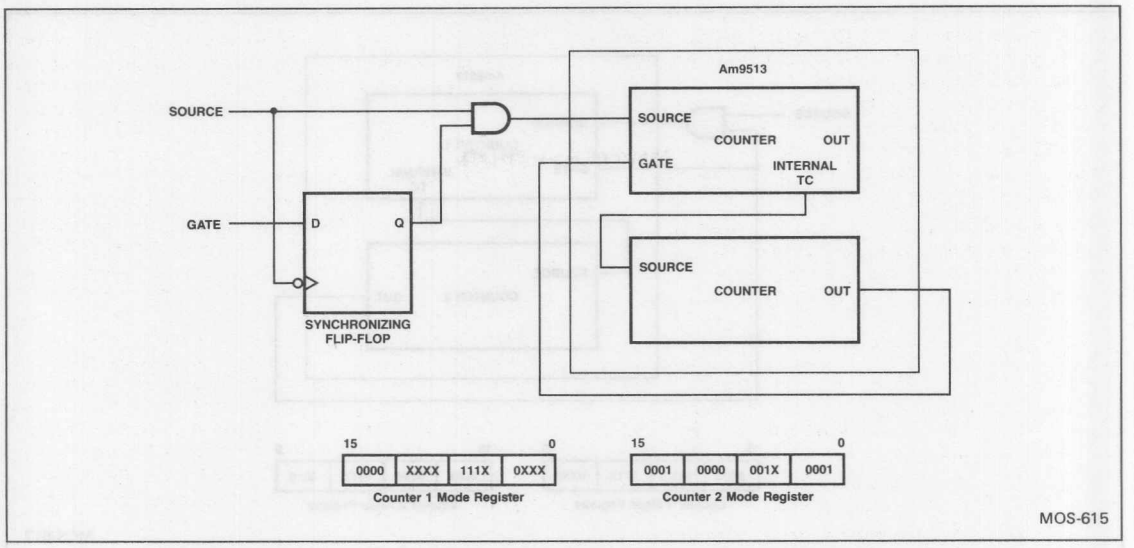


Figure 3-8. Count Down Concatenation with Level Gating

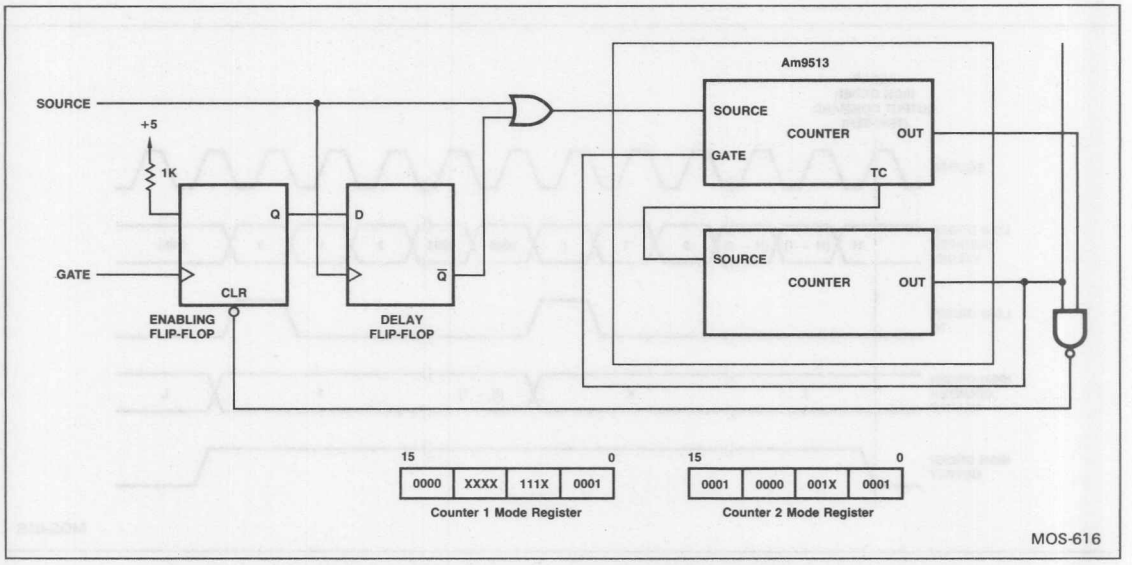


Figure 3-9. Count Down Concatenation with Edge Gating

since the timeout duration remains constant (at (L−1), H for Figure 3-7 and 3926 4178 for the earlier example). To ensure that the counters' first timing cycle has the same timeout duration as subsequent timing cycles, it is important that the high and low order counters be initialized to (L−1) and (H−1) respectively prior to the first timing cycle. Note that if the Counter 1 source period is less than the propagation delay from Counter 1's source through Counter 2's output, through the two flip-flops to the OR gate, then the low order counter's contents at the end of a count cycle may be offset by a few counts. In such cases, the value used to initialize the counters should be similarly offset.

The previous count down concatenation examples have assumed the counters are to count repetitively. To add count once capability to a count down configuration, the high order counter should be programmed to generate a TC Toggled output waveform. This output should be used to gate source pulses through an AND gate into the low order counter, as shown in Figure 3-10. The count cycle will now appear as shown in Figure 3-11. Note that when the counters stop, the high order counter's output will be low and the low order counter's contents will be 9999. To reset the counters for another timing cycle, a LOAD command should be issued to the low order counter, which will

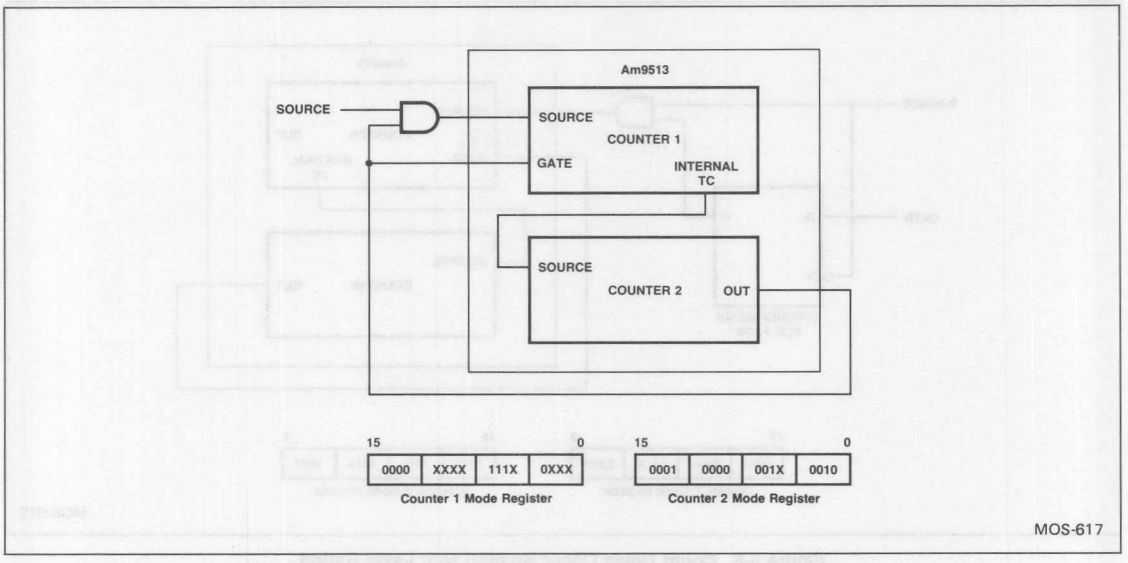


Figure 3-10. Count Down Concatenation with Count Once Feature

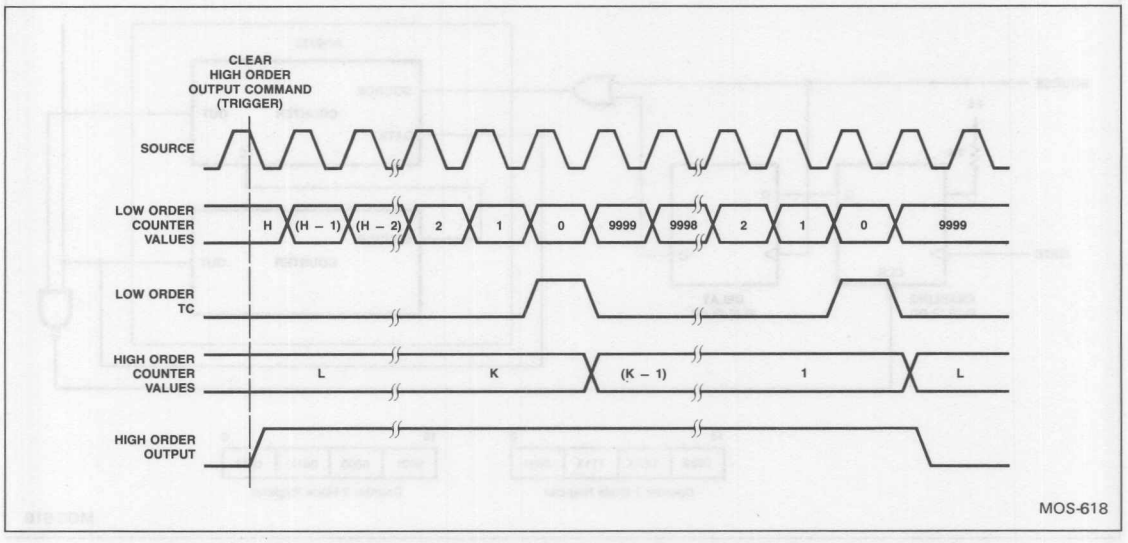


Figure 3-11. Timing Waveforms for Count Down Concatenation with Count Once Feature

reload from the Hold register. The output of the high order counter can now be set to enable counting. Level gating can be added to count-once, count down concatenation by using a 3-input AND gate and driving the third input with an external level gate signal. Count-once, count down concatenation with edge gating can be achieved with the circuit shown in Figure 3-12. The flip-flop is set by an external synchronous gate edge; it is cleared at the end of the count cycle when Counter 2's TC Toggled output goes low.

The concatenation examples presented so far have used two counters to create a 32-bit effective count length. These configurations can be extrapolated to concatenate 3 or more counters to any desired length. Other concatenation variations adventuresome users may wish to investigate are those that use the Alarm registers on Counter 1 and 2 to generate unusual count sequences. Since these Alarm register configurations usually add much complexity for only a limited increase in functionality they are not discussed in this manual.

Saving Concatenated Count Values

The contents of concatenated counters may be read by issuing a SAVE command to the appropriate counters, which will transfer the current counter contents into the counter's Hold registers.

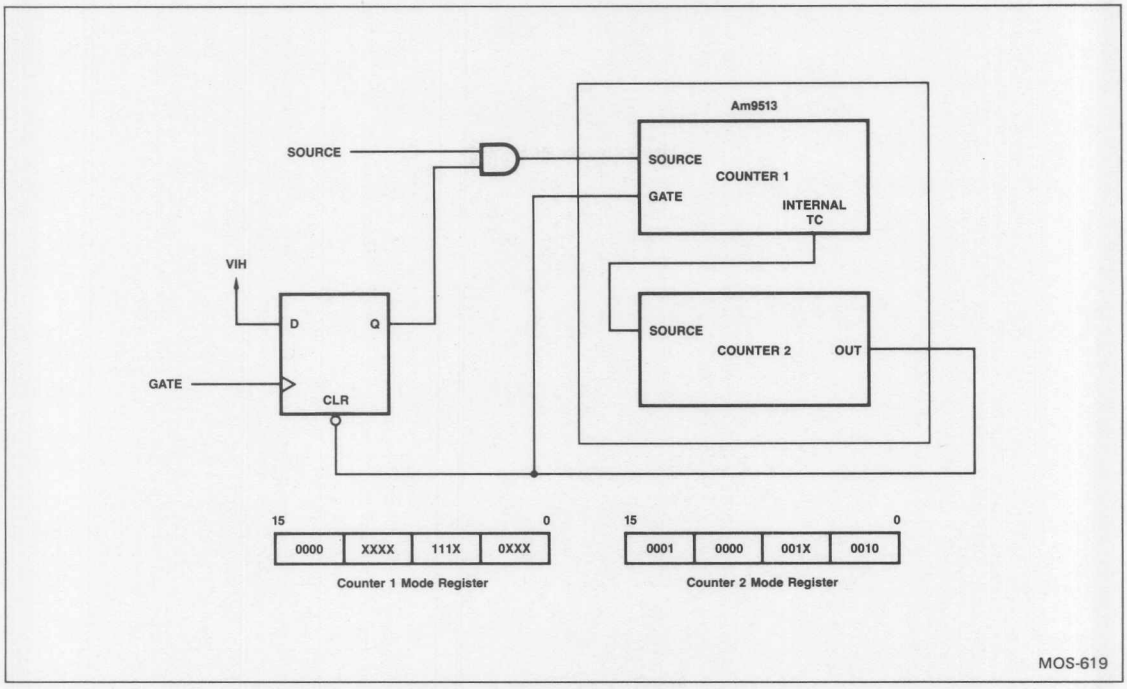
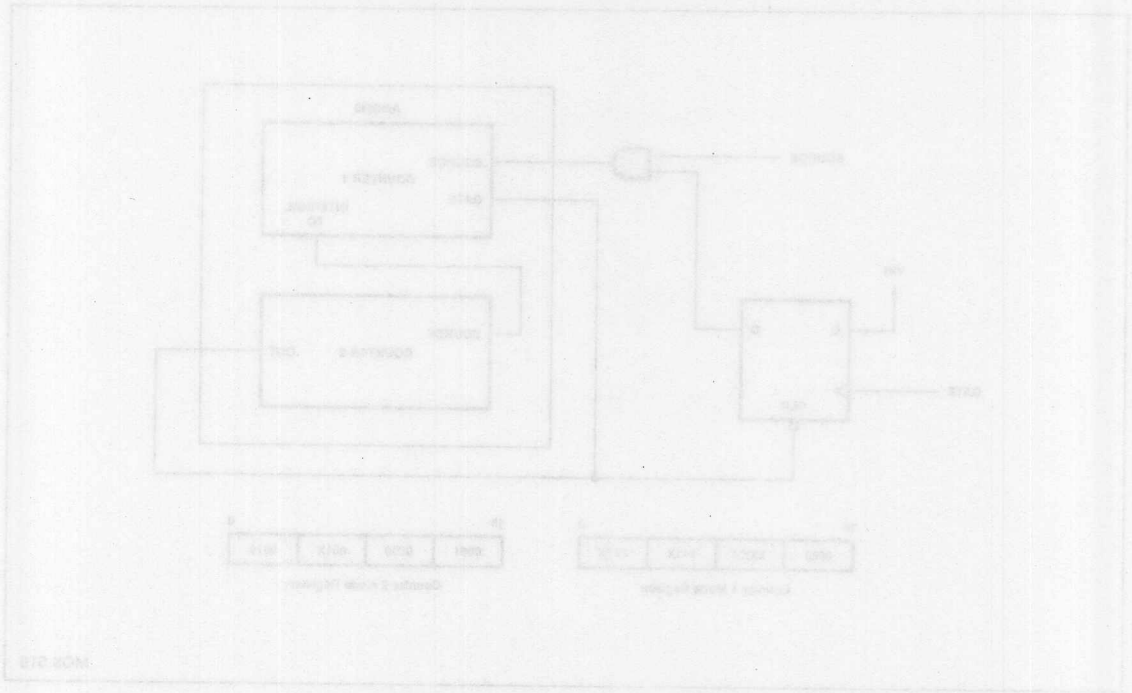


Figure 3-12. Count Down Concatenation with Edge Gating and Count Once Feature

(Since in count down concatenation the Hold register is used to generate the count sequence, in many such applications it may not be feasible to save the low-order counter.) Because the count ripples between concatenated counters, the possibility exists that a SAVE command will be issued after the low order counter increments/decrements but before the carry/borrow ripples through to the high order counter, resulting in an incorrect value being saved in the high order counter's Hold register. The user can protect against this by examining the contents of the low order counter's Hold register immediately after issuing the SAVE command. If the Hold register is equal to the value that would have been expected immediately following generation of a carry/borrow signal, this indicates that the high order value saved is suspect. A new SAVE command should therefore be issued to the high order counter to save a correct count. By the time the low order Hold register contents are read and tested, and a new SAVE command is issued, the high order counter's contents will be stable. The "Time-of-Day" chapter discusses these considerations with respect to Time-of-Day accumulation, and includes a representative software listing.



Chapter 4
Time-of-Day Counting

Chapter 4
Time-of-Day Counting

Time-of-Day Counting with the Am9513

Time-of-day (TOD) counting in the Am9513 is controlled by Master Mode register bits MM0 and MM1. When these bits are set to 01, 10 or 11, logic on Counters 1 and 2 is enabled to cause the counters to roll over at the counts required for Time-of-Day accumulation.

Figure 4-1 shows the format of the 24 hour clock. The two high order decades of Counter 2 contain the hours digits and can hold a maximum count of 23. The two low order Counter 2 decades indicate minutes and will hold values up to 59. The three most significant Counter 1 decades count up to 59.9 and are used to accumulate seconds and tenths of seconds. The low order Counter 1 decade is used to prescale the input frequency to generate tenth-of-second periods to the next higher counter decade.

Figure 4-2 shows the Counter 1 input frequencies supported. Note that the setting of Master Mode register bits MM0 and MM1 selects the optional prescaling factor used in Counter 1's least significant decade. The 50Hz (MM0 = 0, MM1 = 1) option and 60Hz (MM0 = 1, MM1 = 0) option permit AC powerline frequency sources to be used. When the 100Hz (MM0 = 1, MM1 = 1) option is used, the least significant decade of Counter 1 accumulates time in hundredths of a second (tens of milliseconds). Many convenient frequency sources, including the on-chip oscillator, may be used to drive the TOD clock.

Generating Time-of-Day Reference Frequencies

Bits CM11-CM8 in Counter Mode register 1 select the source input used to drive the Counter 1 Time-of-Day circuitry. The variety of inputs offered provides a number of easily implemented frequency sources.

If a 1MHz crystal is connected to the X1 and X2 oscillator inputs, and the BCD prescaling mode is selected (MM15 = 1), a 100Hz signal will be generated on internal signal F5. This F5 source can be selected as the count source for Counter 1 by setting Counter 1 Mode register bits CM11-CM8 to 1111.

As shown in Figure 4-3, crystals between 1 and 10MHz, in 1MHz increments, may also be used. The frequency prescaler is used with BCD scaling mode (MM15 = 1) to divide the crystal's frequency down to a multiple of 100Hz on F5. This F5 signal is then selected by Master Mode bits MM7-MM4 to drive the FOUT pin.

The FOUT scaler, programmed by bits MM11-MM8, is set to generate a 100Hz FOUT signal. The FOUT pin is externally strapped to one of SRC1-SRC5 or GATE1-GATE5. Counter 1 Mode register bits CM11-CM8 are then used to select the driven input pin.

If a spare counter is available, it can be used to prescale almost any crystal input frequency to produce an output rate appropriate for Counter 1's Time-of-Day input. Counter 5 is the easiest to use for this prescaling since its output appears as the TCN−1 input to Counter 1. This means no external strapping is required for the prescaled signal; all that is required is to set the Counter 1 Mode register bits CM11-CM8 to 0000 to select the TCN−1 input.

One readily available, low-cost crystal useful in this configuration is the 3.579545MHz TV color burst crystal. When divided by 59659, a 60Hz output is generated. Figure 4-4 shows a sample configuration using this crystal. Counter 5's Mode register should be set to use F1 as the source, to count down repetitively, to reload from the Load register and to disable special gating. The Gating Control Field should be set to "No Gating." Since internal concatenation is used, the Counter 5 Output Control Field setting is not relevant.

Note that in BCD mode the maximum counter value is 10000 decimal. If Counter 5 is to divide by 59659 decimal, the counter must operate in binary mode to get a range of 65535. Accordingly, binary counting should be selected in Counter 5's Mode register, and the Load register should be loaded with the binary representation of 59659 decimal (E90B hex). Figure 4-4 shows the required Counter 5 Load and Mode register configurations.

Initializing to Current Time-of-Day

The Time-of-Day circuitry requires a special initialization sequence. The following steps MUST be performed in the order given.

The first step is to set the Master Mode register and then the Counter Mode registers to the desired values. The Master Mode bits controlling Time-of-Day are MM0 and MM1; these have been discussed earlier. The user may also set Master Mode bits MM2 and MM3 at this time if the alarm feature is to be used.

The Counter 1 Mode register should be set to select the desired source input. For most real-time applications, the Gating Control field will be set for "No Gating," although edge and level gating

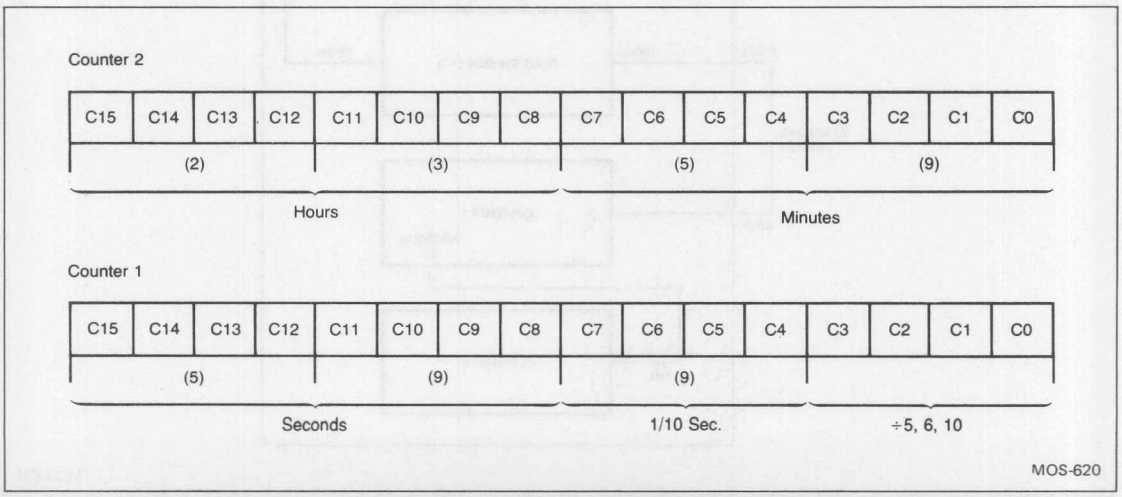


Figure 4-1. Time-of-Day Configuration

MM1	MM0	Configuration
0	0	TOD disabled
0	1	TOD, 50Hz input
1	0	TOD, 60Hz input
1	1	TOD, 100Hz input

Figure 4-2. Time-of-Day Control Options

can be used in Time-of-Day mode. For general purpose time-keeping, the Count Control field should be set to disable special gate; reload from Load; count up; BCD counting; and count repetitively. To generate special functions, the user may select other settings for these bits with the exception of the BCD bit, Time-of-Day counting requires that the counter be set for BCD operation. The Output Control field setting does not affect the Time-of-Day operation since the Counter 1 output will be internally concatenated to the Counter 2 input.

Counter 2's Mode register should be set to the same as Counter 1's with the exception that the Count Source Selection field for Counter 2 should be set to active-high counting from TCN−1, to enable internal concatenation.

The second step is to initialize Counters 1 and 2 to a value of all zeros. This is done by setting Load registers 1 and 2 to 0 and transferring their contents into Counters 1 and 2. Loading the counter with zeros conditions the Time-of-Day count circuitry and must follow the setting of the Master Mode register and Counter Mode registers. If auto-sequencing is being used, each counter's Load register can be set to zero after the Counter's Mode register is loaded. It is important, however, that the Master Mode register be loaded first, and that the Counter Mode registers be loaded before zeros are transferred into the counter. Note that the Master Mode register contents may be changed after this step, providing MM0 and MM1 are not altered; if these are changed, the Time-of-Day circuitry must be reconditioned.

Step 3 of the initialization process involves setting Load registers 1 and 2 to the current time and transferring this into the counters. The format used for the time is that given in Figure 4-1. The user must ensure that the time loaded does not set any of the decades to an illogical value. In particular, no decades should be set to A (hex) through F (hex); the high order decade of Counter 1 and the next-to-least-significant decade in Counter 2 should be 0 through 5; the two high order decades of Counter 2 should be 00 through 23 (BCD); and the low order decade of Counter 1 should be 0. Note also that the Am9513 uses a 24 hour clock, so PM times on a 12 hour clock must have a 12 hour bias added.

The fourth step of the initialization process is to set Load registers 1 and 2 to all 0s. Counter 1 generates a TC and reloads itself from the Load register on the count source edge after reaching 59.99 seconds (100Hz input assumed). Similarly Counter 2 generates a TC on the count source edge after 23:59. By setting the Load registers to 0, the user ensures that the counter will "roll over" to the correct time at TC. For example, a time of 23:59:59.99 will roll over to 00:00:00.00 at midnight.

The final initialization step is to start the counters by writing the "Arm Counters 1 and 2" command (FF23 hex) to the Am9513's Command register. In most Time-of-Day applications, no significant error will be introduced by the delay from the entry of the current time into the counters to the Arming of the counters. This is partly due to the maximum resolution of 10 milliseconds in the Time-of-Day circuitry. In high precision applications, which may use a frequency prescaler for added precision, the user may wish to load a time somewhat later than the current time and delay arming the counter until the current time matches the loaded time.

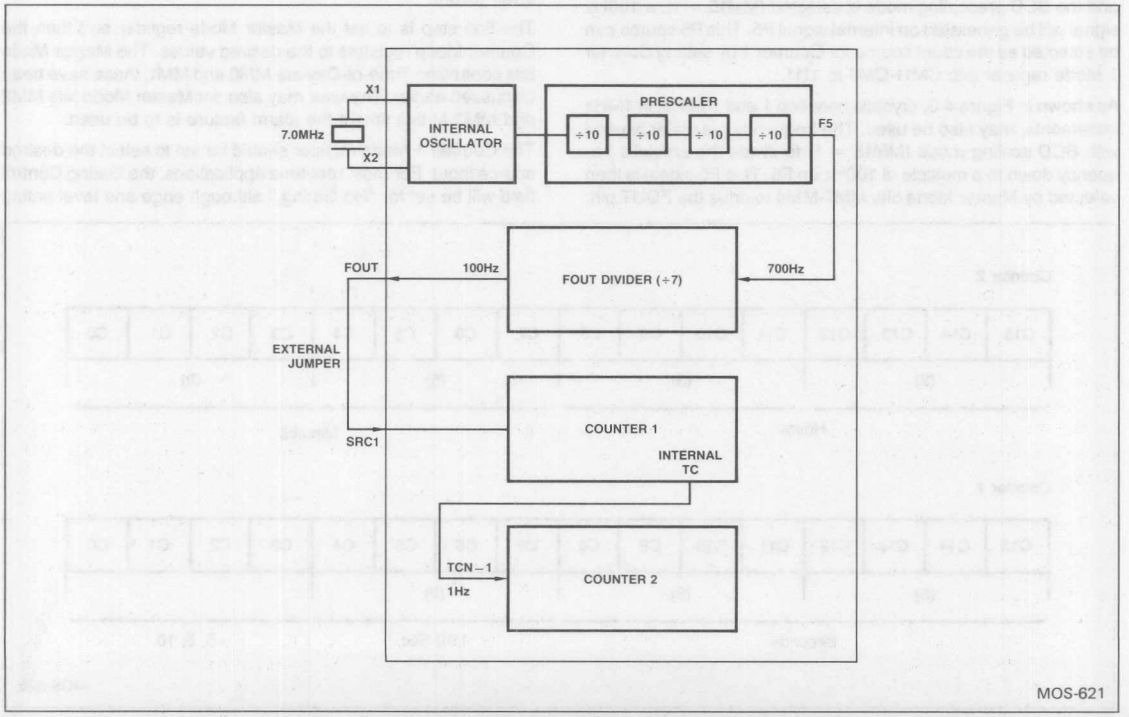


Figure 4-3. Time-of-Day Counting with a 7MHz Crystal

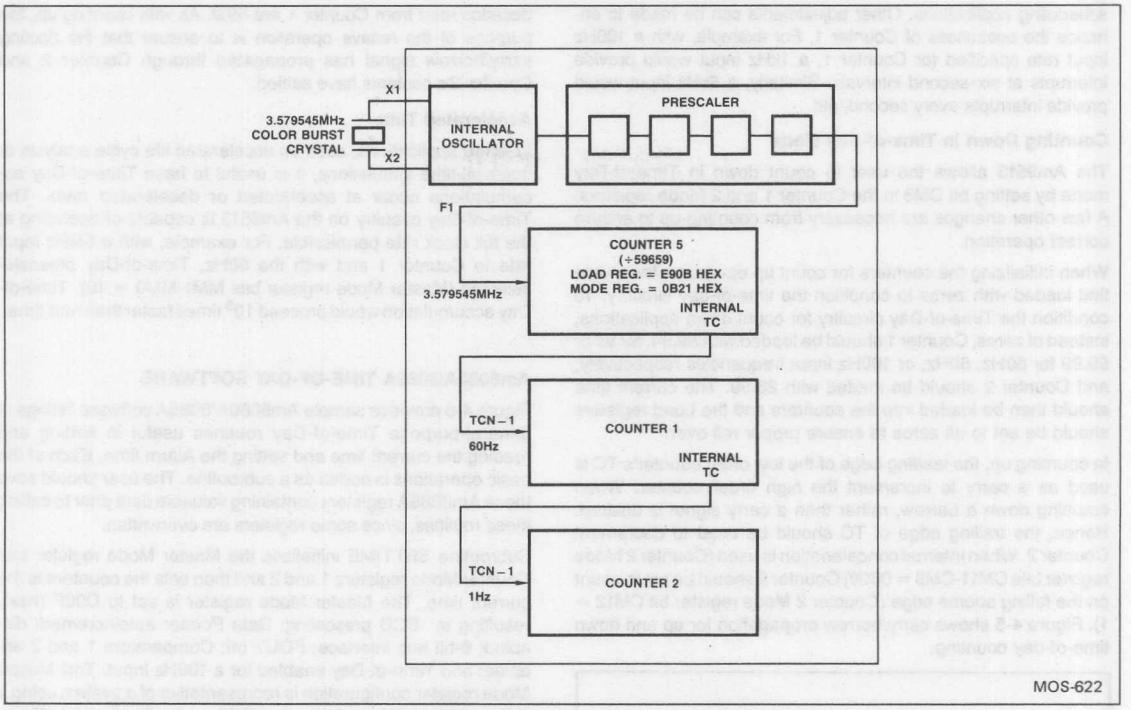


Figure 4-4. Time-of-Day Counting with a Color Burst Crystal

Reading the Current Time

The user may read the current time from the Am9513 by issuing the "Save Counters 1 and 2" command (FFA3 hex) to the Am9513. This causes the contents of Counters 1 and 2 to be transferred into Hold registers 1 and 2. If the user were to now read the Hold register contents, an incorrect time may be read. This is because although Counters 1 and 2 are both synchronous counters, the TC output from Counter 1 ripples through to increment Counter 2. If the save operation occurred just after Counter 1 incremented but before Counter 2 incremented, the value stored in Hold register 2 would be in error. (This consideration should not be confused with the somewhat different points discussed in Appendix A). The user can easily protect against this by examining the contents of Hold register 1. If they are 0, a TC may have just occurred and Hold register 2 may have an erroneous value. Accordingly, if Hold register 1 contains 0, the "Save Counter 2" command (FFA2 hex) should be executed to resave Counter 2. By the time this test for 0 is performed, and Counter 2 is resaved, any rippling carry will have propagated through Counter 2.

Setting the Alarm Time

Master Mode register bits MM2 and MM3 control the Comparators associated with Counters 1 and 2. When a Comparator is enabled, its output is substituted for the normal counter output on the associated OUT1 and OUT2 pins. The polarity definition for the Comparator output will depend on the active-high or active-low definition as programmed in the appropriate Counter Mode register. Once the Comparator output is true, it will remain so until the count changes and the comparison therefore goes false.

The two Comparators can always be used individually in any operating mode. One special case occurs when the time-of-day option is involved and both Comparators are enabled. The operation of Comparator 2 will then be conditioned by Comparator 1 so that a full 32-bit comparison must be true in order to generate a true signal on OUT2. OUT1 will continue, as usual, to reflect the state of the 16-bit comparison between Alarm register 1 and Counter 1.

In some systems, the Alarm output on pin OUT2 might be used to generate an interrupt request to the host CPU. When changing the Alarm register values on a system such as this, the Alarm interrupt should be ignored by the CPU while the Alarm registers are being reloaded, since spurious comparisons may be generated during the reloading process.

Other Time-of-Day Variations

So far this discussion has assumed that Counters 1 and 2 are operated together. The user may, however, elect to drive Counter 2 from a source other than the Counter 1 output, to permit independent accumulation of seconds (Counter 1) and hours/minutes (Counter 2). Note that although Counters 1 and 2 may be independently operated, the Time-of-Day enable bits (MM0 and MM1) are such that either both or neither of the two counters are in Time-of-Day mode; the user cannot set only one counter for Time-of-Day counting.

Separating the two counters would permit Counter 2 to keep track of hours and minutes in real-time, and Counter 1 to time events up to 60 seconds long. The TC output from Counter 1 might perhaps be used to interrupt the CPU. Note that in normal, concatenated Time-of-Day operation, the Counter 1 TC may be used to generate interrupts every minute, a feature useful in real-time

scheduling applications. Other adjustments can be made to enhance the usefulness of Counter 1. For example, with a 100Hz input rate specified for Counter 1, a 1kHz input would provide interrupts at six-second intervals. Similarly, a 6kHz input would provide interrupts every second, etc.

Counting Down in Time-of-Day Mode

The Am9513 allows the user to count down in Time-of-Day mode by setting bit CM3 in the Counter 1 and 2 Mode registers. A few other changes are necessary from counting-up to ensure correct operation.

When initializing the counters for count up operation, they were first loaded with zeros to condition the time-of-day circuitry. To condition the Time-of-Day circuitry for count down applications, instead of zeros, Counter 1 should be loaded with 59.94, 59.95 or 59.99 for 50Hz, 60Hz, or 100Hz input frequencies respectively, and Counter 2 should be loaded with 23.59. The current time should then be loaded into the counters and the Load registers should be set to all zeros to ensure proper roll over.

In counting up, the leading edge of the low order counter's TC is used as a carry to increment the high order counter. When counting down a borrow, rather than a carry signal is desired. Hence, the trailing edge of TC should be used to decrement Counter 2. When internal concatenation is used (Counter 2 Mode register bits CM11-CM8 = 0000) Counter 2 should be set to count on the falling source edge (Counter 2 Mode register bit CM12 = 1). Figure 4-5 shows carry/borrow propagation for up and down time-of-day counting.

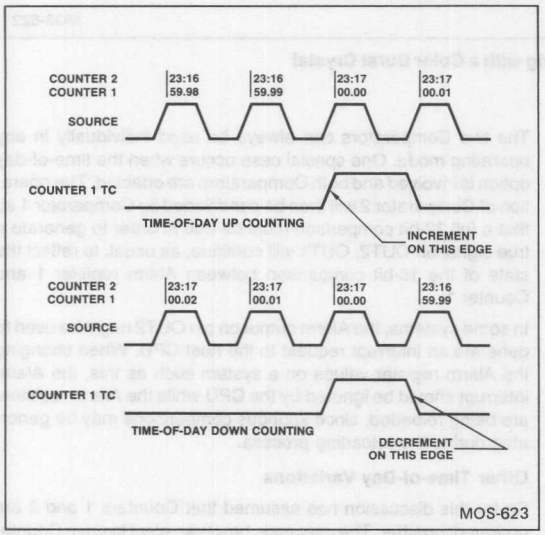


Figure 4-5. Timing Waveforms for Time-of-Day Counting

As with counting up, an incorrect value can be read from the counters if a Save operation is performed while a carry/borrow is rippling between Counters 1 and 2. In count up mode, erroneous readings are prevented by resaving Counter 2 when the value read from Counter 1 was 0. In count down, Counter 2 should be resaved if the value read from Counter 1 indicates a borrow was just generated. This value depends on the input scaler used. For 50Hz, it is 59.94; for 60Hz it is 59.95; for 100Hz it is 59.99. For general-purpose software routines the user can mask out the last decade and resave Counter 2 whenever the upper 3 decades read from Counter 1 are 59.9. As with counting up, the purpose of the resave operation is to ensure that the rippling carry/borrow signal has propagated through Counter 2 and Counter 2's contents have settled.

Accelerated Time

In some applications, such as accelerated life cycle analysis or non-real-time simulations, it is useful to have Time-of-Day accumulations occur at accelerated or decelerated rates. The Time-of-Day circuitry on the Am9513 is capable of operating at the full clock rate permissible. For example, with a 6MHz input rate to Counter 1 and with the 60Hz, Time-of-Day prescaler selected (Master Mode register bits MM1-MM0 = 10), Time-of-Day accumulation would proceed 10^5 times faster than real time.

Am8080A/8085A TIME-OF-DAY SOFTWARE

Figure 4-6 provides sample Am8080A/8085A software listings of general-purpose Time-of-Day routines useful in setting and reading the current time and setting the Alarm time. Each of the basic operations is coded as a subroutine. The user should save those Am8085A registers containing valuable data prior to calling these routines, since some registers are overwritten.

Subroutine SETTIME initializes the Master Mode register and Counter Mode registers 1 and 2 and then sets the counters to the current time. The Master Mode register is set to D00F (hex), resulting in: BCD prescaling; Data Pointer autoincrement disabled; 8-bit bus interface; FOUT off; Comparators 1 and 2 enabled; and Time-of-Day enabled for a 100Hz input. This Master Mode register configuration is representative of a system using a 1MHz crystal on X1 and X2; since BCD prescaling is used, F5 can drive counter 1 with the required 100Hz input signal.

Each Counter Mode register is set for BCD repetitive up counting, with no gating and the special gate disabled. Reloads are set to occur from the Load register only. Counter 1 uses F5 as a count source and Counter 2 uses internal (TCN−1) concatenation. Counter 1's output is set to the high-impedance state. If the comparators had been disabled, a TC output could have been selected for Counter 1 to generate interrupts every minute. (Recall that when the comparators are enabled, the normal TC or toggle output is disabled.) Accordingly, in this sample configuration OUT1, if enabled, would reflect the output of Comparator 1. Counter 2's output is set to active-high TC, but since both Comparators 1 and 2 are enabled in Time-of-Day mode, the output will indicate when a 32-bit match occurs.

Following Counter Mode register initialization, Counters 1 and 2 are loaded with zero to condition the Time-of-Day count circuitry. The current time, stored in variable TIME is then loaded into the counters. Note the data format used for TIME is hours, minutes, seconds and tenths-of-seconds. Finally, the Load registers are reset to zero and the counter is armed.

The second subroutine, ALARMSET, sets the Counter 1 and 2 Alarm registers to the Alarm time stored in variable ALARM. The counter outputs may generate spurious compares while the Alarm registers are being loaded; the subroutine assumes the desired Alarm time has been stored in variable ALARM with the same format used for variable TIME. The subroutine also assumes that Master Mode register bits MM2 and MM3 have been previously set to '1' to enable the comparator circuitry.

Subroutine READTIME reads the time from Counters 1 and 2 and stores it in variable TIME. As discussed previously, special precautions are required when reading the time to avoid saving erroneous values generated by the ripple carry. The subroutine provides the required safeguards.

```
 1:   0100                      ORG 100H
 2:                             ;
 3:                             ;EQU'S FOR AM9513 TIME-OF-DAY
 4:                             ;
 5:   0012 =          CMDPRT    EQU   12H      ;COMMAND PORT
 6:   0010 =          DATAPRT   EQU   10H      ;DATA PORT
 7:                             ;
 8:                             ;
 9:                             ;
10:                             ;SUBROUTINE SETTIME SETS CURRENT
11:                             ;TIME USING DATA STORED IN VARIABLE TIME
12:                    SETTIME:
13:                             ;SET MASTER MODE REGISTER
14:                             ;
15:   0100 3E17                 MVI A,017H       ;SET DATA POINTER TO
16:   0102 D312                 OUT CMDPRT       ;MASTER MODE REGISTER
17:   0104 3E0F                 MVI A,0FH        ;LOWER BYTE
18:   0106 D310                 OUT DATAPRT
19:   0108 3ED0                 MVI A,0D0H       ;UPPER BYTE
20:   010A D310                 OUT DATAPRT
21:                             ;
22:                             ;SET COUNTER 1 MODE REG.
23:                             ;
24:   010C 3E01                 MVI A,01H        ;SET DATA POINTER TO
25:   010E D312                 OUT CMDPRT       ;COUNTER 1 MODE REG
26:   0110 3E3C                 MVI A,03CH       ;LOWER BYTE
27:   0112 D310                 OUT DATAPRT
28:   0114 3E0F                 MVI A,0FH        ;UPPER BYTE
29:   0116 D310                 OUT DATAPRT
30:                             ;
31:                             ;SET COUNTER 2 MODE REG.
32:                             ;
33:   0118 3E02                 MVI A,02H        ;SET DATA POINTER TO
34:   011A D312                 OUT CMDPRT       ;COUNTER 2 MODE REG
35:   011C 3E39                 MVI A,039H       ;LOWER BYTE
36:   011E D310                 OUT DATAPRT
37:   0120 3E00                 MVI A,0H         ;UPPER BYTE
38:   0122 D310                 OUT DATAPRT
39:                             ;
40:                             ;CLEAR COUNTERS 1 AND 2
41:                             ;
42:   0124 3E09                 MVI A,09H        ;SET DATA POINTER TO
43:   0126 D312                 OUT CMDPRT       ;LOAD REG 1
44:   0128 3E00                 MVI A,0H
45:   012A D310                 OUT DATAPRT      ;LOWER BYTE
46:   012C D310                 OUT DATAPRT      ;UPPER BYTE
47:                             ;
48:   012E 3E0A                 MVI A,0AH        ;SET DATA POINTER TO
49:   0130 D312                 OUT CMDPRT       ;LOAD REG 2
50:   0132 3E00                 MVI A,0H
51:   0134 D310                 OUT DATAPRT      ;LOWER BYTE
52:   0136 D310                 OUT DATAPRT      ;UPPER BYTE
53:                             ;
54:   0138 3E43                 MVI A,043H       ;TRANSFER CONTENETS OF LOAD
55:   013A D312                 OUT CMDPRT       ;REGS 1 AND 2 INTO COUNTERS
56:                             ;
57:                             ;SET COUNTERS 1 AND 2 TO CURRENT TIME
58:                             ;
```

Figure 4-6. Sample 8080/8085 Time-of-Day Subroutines

```
 59:   013C 3E09              MVI A,09H       ;SET DATA POINTER TO
 60:   013E D312              OUT CMDPRT      ;COUNTER 1 LOAD REG.
 61:   0140 3A7801            LDA TIME+3      ;TENTHS-OF-SECONDS
 62:   0143 D310              OUT DATAPRT
 63:   0145 3A7701            LDA TIME+2      ;SECONDS
 64:   0148 D310              OUT DATAPRT
 65:                         ;
 66:   014A 3E0A              MVI A,0AH       ;SET DATA POINTER TO
 67:   014C D312              OUT CMDPRT      ;COUNTER 2 LOAD REG
 68:   014E 3A7601            LDA TIME+1      ;MINUTES
 69:   0151 D310              OUT DATAPRT
 70:   0153 3A7501            LDA TIME        ;HOURS
 71:   0156 D310              OUT DATAPRT
 72:                         ;
 73:   0158 3E43              MVI A,043H      ;TRANSFER CONTENTS OF LOAD
 74:   015A D312              OUT CMDPRT      ;REGS 1 AND 2 INTO COUNTERS
 75:                         ;
 76:                         ;SET LOAD REG 1 AND 2 TO 0
 77:                         ;
 78:   015C 3E09              MVI A,09H       ;SET DATA POINTER TO
 79:   015E D312              OUT CMDPRT      ;LOAD REG 1
 80:   0160 3E00              MVI A,0H
 81:   0162 D310              OUT DATAPRT     ;LOWER BYTE
 82:   0164 D310              OUT DATAPRT     ;UPPER BYTE
 83:                         ;
 84:   0166 3E0A              MVI A,0AH       ;SET DATA POINTER TO
 85:   0168 D312              OUT CMDPRT      ;LOAD REG 2
 86:   016A 3E00              MVI A,0H
 87:   016C D310              OUT DATAPRT     ;LOWER BYTE
 88:   016E D310              OUT DATAPRT     ;UPPER BYTE
 89:                         ;
 90:                         ;ARM COUNTERS 1 AND 2 TO START COUNT
 91:                         ;
 92:   0170 3E23              MVI A,023H      ;ARM 1 AND 2 COMMAND
 93:   0172 D312              OUT CMDPRT
 94:                         ;
 95:   0174 C9                RET
 96:                         ;
 97:                         ;
 98:   0175 00000000  TIME:   DB      0,0,0,0 ;CURRENT TIME, IN FORMAT
 99:                                     ;HOURS, MINUTES, SECONDS,
100:                                     ;TENTHS-OF-SECONDS. NOTE LOWER
101:                                     ;DECADE OF TENTHS-OF-SECONDS DIGIT
102:                                     ;MUST BE 0.
103:   0179 00000000  ALARM:  DB      0,0,0,0 ;ALARM TIME, SAME FORMAT
104:                                     ;AS TIME.
105:                        ;
106:                        ;
107:                ALARMSET:       ;SETS ALARM TO VALUE
108:                                ;STORED IN VARIABLE ALARM
109:                        ;
110:   017D 3E07              MVI A,07H       ;SET DATA POINTER TO
111:   017F D312              OUT CMDPRT      ;ALARM REG 1
112:   0181 3A7C01            LDA ALARM+3     ;TENTHS-OF-SECONDS
113:   0184 D310              OUT DATAPRT
114:   0186 3A7B01            LDA ALARM+2     ;SECONDS
115:   0189 D310              OUT DATAPRT
116:                         ;
117:   018B 3E0F              MVI A,0FH       ;SET DATA POINTER TO
```

Figure 4-6. Sample 8080/8085 Time-of-Day Subroutines (Cont.)

```
118:   018D D312              OUT CMDPRT      ;ALARM REG 2
119:   018F 3A7A01            LDA ALARM+1     ;MINUTES
120:   0192 D310              OUT DATAPRT
121:   0194 3A7901            LDA ALARM       ;HOURS
122:   0197 D310              OUT DATAPRT
123:   0199 C9                RET
124:                          ;
125:                          ;
126:                          ;
127:                 READTIME:       ;READS CURRENT TIME AND
128:                                 ;STORES IT IN VARIABLE TIME
129:                          ;
130:   019A 3EA3              MVI A,0A3H      ;SAVE COUNTERS 1 AND 2
131:   019C D312              OUT CMDPRT
132:                          ;
133:   019E 3E11              MVI A,011H      ;SET DATA POINTER TO
134:   01A0 D312              OUT CMDPRT      ;COUNTER 1 HOLD REG.
135:   01A2 DB10              IN DATAPRT      ;TENTHS-OF-SECONDS
136:   01A4 47                MOV B,A         ;SAVE IN B REG FOR LATER
137:   01A5 327801            STA TIME+3
138:   01A8 DB10              IN DATAPRT      ;SECONDS
139:   01AA 327701            STA TIME+2
140:   01AD A7                ANA A           ;TEST FOR 0
141:   01AE C2BA01            JNZ CTR2IN
142:   01B1 78                MOV A,B         ;GET TENTHS-OF-SECONDS
143:   01B2 A7                ANA A           ;TEST FOR 0
144:   01B3 C2BA01            JNZ CTR2IN
145:                          ;
146:                          ;COUNTER 1 HAD 0, SAVE CTR 2 AGAIN
147:                          ;
148:   01B6 3EA1              MVI A,0A1H      ;SAVE CTR 2 COMMAND
149:   01B8 D312              OUT CMDPRT
150:                          ;
151:   01BA 3E12       CTR2IN: MVI A,12H      ;SET DATA POINTER TO
152:   01BC D312              OUT CMDPRT      ;COUNTER 2 HOLD REG
153:   01BE DB10              IN DATAPRT      ;MINUTES
154:   01C0 327601            STA TIME+1
155:   01C3 DB10              IN DATAPRT      ;HOURS
156:   01C5 327501            STA TIME
157:                          ;
158:   01C8 C9                RET
159:                          ;
160:   01C9                   END
```

Figure 4-6. Sample 8080/8085 Time-of-Day Subroutines (Cont.)

A COOKBOOK APPROACH TO TIME-OF-DAY COUNTING

The following steps are given as a simple, easy-to-use guide to the operation of the Time-of-Day Software. At the end of each step is a reference to the line number appearing along the left side of the program listing in Figure 4-6. Note that these programs have been coded to maximize clarity. Use of automatic Data Pointer sequencing and special programming tricks would likely result in more compact code. The program assumes that the Am9513's Command port is located at address 12 (hex) and that the Data port is at address 10 (hex).

SETTING THE CURRENT TIME

Subroutine SETTIME in the software listing sets the current time and configures the Master Mode register and Counter Mode register 1 and 2.

1. Set the Master Mode register as follows:

   a) Set the Data Pointer register to point to the Master Mode register by writing 17 (hex) to the Am9513 Command port (Lines 15-16).

b) Write XXXX AA11 (binary) to the Am9513 Data port to set the low order byte of the Master Mode register (Lines 17-18).

X = don't care bit
A = set to '1' for Alarm function; set to '0' if Alarm will not be used.

c) Write 110X XXXX (binary) to the Am9513 Data port to set the high order byte of the Master Mode register (Lines 19-20).

2. Set Counter 1's Mode register as follows:
   a) Set the Data Pointer to the Counter 1 Mode register by writing 01 (hex) to the Command port (Lines 24-25).
   b) Write 3C (hex) to the Data port. This is the lower byte of the Counter 1 Mode register (Lines 26-27).
   c) Write 0F (hex) to the Data port. This is the upper byte of the Counter 1 Mode register (Lines 28-29).
3. Set Counter 2's Mode register as follows:
   a) Set the Data Pointer to Counter 2's Mode register by writing 02 (hex) to the Command port (Lines 33-34).
   b) Set the low byte of Counter 2's Mode register by writing 39 (hex) to the Am9513 Data port (Lines 35-36).
   c) Write 00 (hex) to the Data port to set the upper byte (Lines 37-38).
4. Counters 1 and 2 must now be loaded with 0 to condition the Time-of-Day circuitry. Set Load register 1 to 0 as follows:
   a) Set the Data Pointer register to point to Load register 1 by writing 09 (hex) to the Am9513 Command port (Lines 42-43).
   b) Write 00 (hex) to the Data port to set the lower byte of the Load register (Lines 44-45).
   c) Write 00 (hex) to the Data port to set the upper byte (Line 46).
5. Set Load register 2 to 0 as follows:
   a) Set the Data Pointer register to point to Load register 2 by writing 0A (hex) to the Am9513 Command port (Lines 48-49).
   b) Write 00 (hex) to the Data port to set the lower byte of the Load register (Lines 50-51).
   c) Write 00 (hex) to the Data port to set the upper byte (Line 52).
6. Transfer the contents of Load registers 1 and 2 into Counters 1 and 2 by writing the "Load Counters 1 and 2" command (43 hex) to the Am9513 Command port (Lines 54-55).
7. Set the current time's seconds and tenths-of-seconds into Load register 1 as follows:
   a) Set the Data Pointer register to point to Counter 1's Load register by writing 09 (hex) to the Am9513's Command port (Lines 59-60).
   b) Write the current time's tenths-of-seconds to the Data port in the format T0 (hex), where T is a BCD number between 0 to 9 representing tenths of seconds. The low decade, used by the Time-of-Day prescaler, will usually be set to 0. (If a 100Hz input frequency is selected the low decade may be set to the current time's tens-of-milliseconds value.) (Lines 61-62)
   c) Write the current time's seconds to the Data port. This should be a number between 0 and 59 (decimal) in BCD format (Lines 63-64).
8. Set the current time's hours and minutes, using a 24 hour clock, into Load register 2 as follows:
   a) Set the Data Pointer register to point to Counter 2's Load register by writing 0A (hex) to the Am9513's Command port (Lines 66-67).
   b) Write the current time's minutes to the Data port. This number must be between 00 and 59 (decimal) in BCD format (Lines 68-69).
   c) Write the current time's hours to the Data port. This number must be between 00 and 23 (decimal) in BCD format (Lines 70-71).
9. Load Counters 1 and 2 with the current time by writing the "Load Counters 1 and 2" command (43 hex) to the Am9513's Command port (Lines 73-74).
10. Set Load register 1 to 0 by repeating Step 4 (Lines 78-82).
11. Set Load register 2 to 0 by repeating Step 5 (Lines 84-88).
12. Start the counters by writing the "Arm Counters 1 and 2" command (23 hex) to the Am9513's Command port (Lines 92-93).

SETTING THE ALARM TIME

1. Set Alarm register 1 to the seconds and tenths-of-seconds Alarm time as follows:
   a) Set the Data Pointer register to point to Alarm register 1 by writing 07 (hex) to the Am9513's Command port (Lines 110-111).
   b) Write the tenths-of-seconds Alarm time to the Am9513's Data port in the format T0 (hex) where T is a number between 0 and 9 in BCD format representing tenths-of-seconds. The lower decade will usually be set to 0. (When a 100Hz input frequency is selected, the lower decade may be set to the Alarm times tens-of-millseconds.) (Lines 112-113)
   c) Write the seconds component of the Alarm time to the Data port. This should be a BCD number of 0 through 59 (decimal) (Lines 114-115).
2. Set Alarm register 2 to the hours and minutes Alarm time as follows:
   a) Set the Data Pointer register to point to Alarm register 2 by writing 0F (hex) to the Am9513's command port (Lines 117-118).
   b) Write the minute component of the Alarm time to the Data port. This should be a BCD encoded number of value 0 through 59 (decimal) (Lines 119-120).
   c) Write the hours component of the Alarm time to the Data port. This should be a BCD encoded number of value 0 through 23 (decimal) (Lines 121-122).
3. If Master Mode register bits MM2 and MM3 are not already both 1, set them to 1 now as follows: (These steps are not shown in the software listing.)
   a) Set the Data Pointer register to point to the Master Mode register by writing 17 (hex) to the Am9513's Command port.
   b) Read the lower Master Mode register byte.
   c) Perform a logical OR between this byte and 0C (hex). This sets bits 2 and 3 to 1.
   d) Repeat step a) to reset the Data Pointer register.
   e) Write the resultant byte from step c) to the Master Mode register. This updates the lower byte of the register.

READING THE CURRENT TIME

1. Save the contents of Counters 1 and 2 in Hold registers 1 and 2 by writing the "Save Counters 1 and 2" command (A3 hex) to the Am9513's Command port (Lines 130-131).

2. Read the value saved in Hold register 1 as follows:
   a) Set the Data Pointer register to point to Counter 1's Hold register by writing 11 (hex) to the Am9513's Command port (Lines 133-134).
   b) Read from the Data port to retrieve the low order Hold register byte. The upper decade contains the tenths-of-seconds component of the current time. The lower decade is used by the Time-of-Day prescaler. For 100Hz prescaling, it is tens-of-milliseconds (Lines 135-137).
   c) Read from the Data port again to retrieve the high order byte. This is the seconds component of the current time (Lines 138-139).
3. If the value read from Hold register 1 is 0000, Save Counter 2 again. Otherwise, go to Step 4. Counter 2 may be resaved by writing A2 (hex) to the Command port (Lines 140-149).
4. Read the contents of Hold register 2 as follows:
   a) Set the Data Pointer register to point to Load register 2 by writing 12 (hex) to the Am9513's Command port (Lines 151-152).
   b) Read the minutes component of the current time by reading a byte from the Data port (Lines 153-154).
   c) Read the hours component of the current time by reading another byte from the Data port (Lines 155-156).

SETTIME SOFTWARE USING MACROS

The programming examples given above can be further simplified by using macros. The listing in Figure 4-7 is a macro-coded version of the subroutine SETTIME for the 8080, 8085 and Z80.* This example provides a graphic illustration of the simplicity of usage of the Am9513 macros. The full output code generation of the macroassembler is shown in Figure 4-8 to illustrate the convenience of using macros. This code expansion is slightly different than the code in Figure 4-6 due to different counter starting points and some software efficiencies.

CONSOLE DRIVEN CLOCK RUNS UNDER CP/M

A further example written in C is presented; it provides a simple console-driven clock running under the CP/M* (ver 2.2) compatible AMDOS® operating system. The resolution of the clock is limited to one-second intervals since the Whitesmiths' C compiler unfortunately generates an I/O overhead of such a size that the loading time of the program makes further accuracy pointless.

The C example shown in Figure 4-9 illustrates the two ways of setting the various modes of operation and register values desired. The Master Mode register (master-reg in the listing) is statically initialized since the various field values are known at compile time. This declaration generates just two bytes of code corresponding to the 16 bits of the Master Mode register.

The two Counter Mode registers are dynamically initialized field by field (refer to the set modes procedure in the listing). Actually, static initialization could be used, since the desired field values are known at compile time (e.g., .base = BCD). For the purposes of illustration, however, dynamic initialization is employed.

The program protects against reading the time while a carry is in progress from counter 1 TC to the counter 2 input. Should the value read from counter 1 be zero then the contents of counter 2 are resaved to avoid possible misreading of the time.

*Z80 is a trademark of Zilog, Inc.
CP/M is a trademark of Digital Research, Inc.
AMDOS is a registered trademark of Advanced Micro Devices, Inc.

```
;       Coded for Am8080/Am8085 - also runs on Z80

        INCLUDE B:EQUS.MAC              ; Listed in Appendix 7
        INCLUDE B:8080.MAC              ; Listed in Appendix 3

SETTIME:
        MASTER          TOD_100HZ,ENABLE,ENABLE,F1,0,OF,BUS_8,OF,BCD
        MODE_REG        1,OFF_OC_TC,UP,BCD,MODE_DEF,F5,RISE,NO_GATE
        MODE_REG        2,ACT_HI_TC,UP,BCD,MODE_DEF,TC_NM1,RISE,NO_GATE
        LOAD_REG        1,0
        LOAD_REG        2,0
        LOAD            1,2             ; Enable TOD counting
        LOAD_REG        1,TIME+2,I
        LOAD_REG        2,TIME,I
        LOAD            1,2             ; Set the current time
        LOAD_REG        1,0
        LOAD_REG        2,0             ; Recall repeat reload values needed
        ARM             1,2             ; Time starts now
        RET

                                ; Current time hrs, min, secs, tenths
TIME:   DB      0,0,0,0
```

Figure 4-7. SETTIME Macro Listing for 8080, 8085 and Z80

```
    - Macro expansion of SETTIME.MAC example -

0100'           SETTIME:

                MASTER  TOD_100HZ,ENABLE,ENABLE,F1,0,OF,BUS_8,OF,BCD

0003              +   DLAB        SET     TOD_100HZ
0007              +   DLAB        SET     DLAB OR (ENABLE SHL 2)
000F              +   DLAB        SET     DLAB OR (ENABLE SHL 3)
00BF              +   DLAB        SET     DLAB OR (F1 SHL 4)
00BF              +   DLAB        SET     DLAB OR (0 SHL 8)
10BF              +   DLAB        SET     DLAB OR (OF SHL 12)
10BF              +   DLAB        SET     DLAB OR (0 SHL 13)
50BF              +   DLAB        SET     DLAB OR (OF SHL 14)
D0BF              +   DLAB        SET     DLAB OR (BCD SHL 15)
0100'  3E 17      +               MVI     A,CTRL_GR OR (MASTER_ SHL 3)
0102'  D3 12      +               OUT     A_CTRL
0104'  3E BF      +               MVI     A, LOW DLAB
0106'  D3 10      +               OUT     A_DATA
0108'  3E D0      +               MVI     A, HIGH DLAB
010A'  D3 10                      OUT     A_DATA

                MODE_REG  1,OFF_OC_TC,UP,BCD,MODE_DEF,F5,RISE,NO_GATE

0004              +   DLAB        SET     OFF_OC_TC
000C              +   DLAB        SET     DLAB OR (UP SHL 3)
001C              +   DLAB        SET     DLAB OR (BCD SHL 4)
003C              +   DLAB        SET     DLAB OR (MODE_DEF SHL 5)
0F3C              +   DLAB        SET     DLAB OR (F5 SHL 8)
0F3C              +   DLAB        SET     DLAB OR (RISE SHL 12)
0F3C              +   DLAB        SET     DLAB OR (NO_GATE SHL 13)
010C'  3E 01                      MVI     A,1 OR (MODE_ SHL 3)
010E'  D3 12                      OUT     A_CTRL
0110'  3E 3C                      MVI     A, LOW DLAB
0112'  D3 10                      OUT     A_DATA
0114'  3E 0F                      MVI     A, HIGH DLAB
0116'  D3 10                      OUT     A_DATA

                MODE_REG  2,ACT_HI_TC,UP,BCD,MODE_DEF,TC_NM1,RISE,NO_GATE

0001              +   DLAB        SET     ACT_HI_TC
0009              +   DLAB        SET     DLAB OR (UP SHL 3)
0019              +   DLAB        SET     DLAB OR (BCD SHL 4)
0039              +   DLAB        SET     DLAB OR (MODE_DEF SHL 5)
0039              +   DLAB        SET     DLAB OR (TC_NM1 SHL 8)
0039              +   DLAB        SET     DLAB OR (RISE SHL 12)
0039              +   DLAB        SET     DLAB OR (NO_GATE SHL 13)
0118'  3E 02                      MVI     A,2 OR (MODE_ SHL 3)
011A'  D3 12                      OUT     A_CTRL
011C'  3E 19                      MVI     A, LOW DLAB
011E'  D3 10                      OUT     A_DATA
0120'  3E 00                      MVI     A, HIGH DLAB
0122'  D3 10                      OUT     A_DATA
```

Figure 4-8. Macro Assembler Output

```
                LOAD_REG        1,0

0124'   3E 09                   MVI     A,1 OR (LOAD_ SHL 3)
0126'   D3 12                   OUT     A_CTRL
0128'   3E 00                   MVI     A, LOW 0
012A'   D3 10                   OUT     A_DATA
012C'   3E 00                   MVI     A, HIGH 0
012E'   D3 10                   OUT     A_DATA

                LOAD_REG        2,0

0130'   3E 0A                   MVI     A,2 OR (LOAD_ SHL 3)
0132'   D3 12                   OUT     A_CTRL
0134'   3E 00                   MVI     A, LOW 0
0136'   D3 10                   OUT     A_DATA
0138'   3E 00                   MVI     A, HIGH 0
013A'   D3 10                   OUT     A_DATA

                LOAD            1,2             ; Enable TOD counting

0040              + DLAB        SET     40H
0042              + DLAB        SET     DLAB OR (1 SHL (2-1))
0043              + DLAB        SET     DLAB OR (1 SHL (1-1))
013C'   3E 43                   MVI     A,DLAB
013E'   D3 12                   OUT     A_CTRL

                LOAD_REG        1,TIME+2,I

0140'   3E 09                   MVI     A,1 OR (LOAD_ SHL 3)
0142'   D3 12                   OUT     A_CTRL
0144'   3A 017F'                LDA     TIME+2
0147'   D3 10                   OUT     A_DATA
0149'   3A 0180'                LDA     TIME+2+1
014C'   D3 10                   OUT     A_DATA

                LOAD_REG        2,TIME,I

014E'   3E 0A                   MVI     A,2 OR (LOAD_ SHL 3)
0150'   D3 12                   OUT     A_CTRL
0152'   3A 017D'                LDA     TIME
0155'   D3 10                   OUT     A_DATA
0157'   3A 017E'                LDA     TIME+1
015A'   D3 10                   OUT     A_DATA

                LOAD            1,2             ; Set the current time

0040              + DLAB        SET     40H
0042              + DLAB        SET     DLAB OR (1 SHL (2-1))
0043              + DLAB        SET     DLAB OR (1 SHL (1-1))
015C'   3E 43                   MVI     A,DLAB
015E'   D3 12                   OUT     A_CTRL
```

Figure 4-8. Macro Assembler Output (Cont.)

```
                LOAD_REG        1,0
0160'   3E 09                  MVI     A,1 OR (LOAD_ SHL 3)
0162'   D3 12                  OUT     A_CTRL
0164'   3E 00                  MVI     A, LOW 0
0166'   D3 10                  OUT     A_DATA
0168'   3E 00                  MVI     A, HIGH 0
016A'   D3 10                  OUT     A_DATA
                LOAD_REG        2,0             ; Recall repeat reload
                                                    values needed
016C'   3E 0A                  MVI     A,2 OR (LOAD_ SHL 3)
016E'   D3 12                  OUT     A_CTRL
0170'   3E 00                  MVI     A, LOW 0
0172'   D3 10                  OUT     A_DATA
0174'   3E 00                  MVI     A, HIGH 0
0176'   D3 10                  OUT     A_DATA

                ARM             1,2             ; Time starts now

0020              +  DLAB       SET     20H
0022              +  DLAB       SET     DLAB OR (1 SHL (2-1))
0023              +  DLAB       SET     DLAB OR (1 SHL (1-1))
0178'   3E 23                  MVI     A,DLAB
017A'   D3 12                  OUT     A_CTRL
017C'   C9                            RET

                                                ; Current time hrs, min,
017D'   00 00 00 00             TIME:   DB      0,0,0,0     secs, tenths

                END SETTIME
Macros:

ARM             CLEAR           DISARM          DPS             DRMSAV
FOUT            HOLD_R          LD_ARM          LOAD            LOAD_R
MASTER          MODE_R          N_TYPE          POINT           RESET
SAVE            SET_            STEP            S_MASK          S_TYPE

Symbols:

ACT_HI  0001    ACT_LO  0005    ALARM1  0000    ALARM2  0001
A_CTRL  0012    A_DATA  0010    BCD     0001    BINARY  0000
BUS_16  0001    BUS_8   0000    CTRL_G  0007    DISABL  0000
DLAB    0023    DOWN    0000    ENABLE  0001    F1      000B
F2      000C    F3      000D    F4      000E    F5      000F
FALL    0001    GATE_1  0006    GATE_2  0007    GATE_3  0008
GATE_4  0009    GATE_5  000A    HE_GAT  0006    HL_GAT  0004
HL_NM1  0003    HL_NP1  0002    HL_TC_  0001    HOLD_   0002
HOLD_C  0003    I       0000    LE_GAT  0007    LL_GAT  0005
LOAD_   0001    MASTER  0002    MODE_   0000    MODE_A  0000
MODE_D  0001    MODE_G  0002    MODE_J  0003    MODE_M  0004
MODE_P  0005    MODE_S  0006    MODE_V  0007    NO_GAT  0000
OF      0001    OFF_LO  0000    OFF_OC  0004    ON      0000
RISE    0000    SETTIM  0100'   SRC_1   0001    SRC_2   0002
SRC_3   0003    SRC_4   0004    SRC_5   0005    STATUS  0003
TC_NM1  0000    TC_TOG  0002    TIME    017D'   TOD_10  0003
TOD_50  0001    TOD_60  0002    TOD_OF  0000    UP      0001

No  Fatal error(s)

A>
```

Figure 4-8. Macro Assembler Output (Cont.)

```
/*      This file provides an example usage of the 9513 System Timing
controller device as a console clock. Whitesmiths' C example.
*/
#include "B:9513DEFS.H"                 /* Listed in Appendix 6 */
#include "B:9513RECS.H"                 /* Listed in Appendix 1 */

/*      Record structure definitions - Whitesmiths' C demands initialisers.

        Notice that the master_reg definition generates only 2 bytes of code,
  as would the counter definitions if they were statically initialised.
*/

master_type     master_reg      /* This is a static initialisation      */

                = {TOD_100HZ,DISABLE,DISABLE,F1,7,OFF,BUS_8,OFF,BCD} ;


channel_type    count[6]        /* array of the 5 counters on the 9513  */

                = {0,0,0,0,0,0,0,0,0,   /* dummy entry - index for count1 is 1 */
                   0,0,0,0,0,0,0,0,0,
                   0,0,0,0,0,0,0,0,0,
                   0,0,0,0,0,0,0,0,0,
                   0,0,0,0,0,0,0,0,0,
                   0,0,0,0,0,0,0,0,0
                   } ;


/*      Command constants       */

load    =       0X43 ;          /* LOAD counters 1 and 2        */
arm     =       0X23 ;          /* ARM counters 1 and 2         */
save    =       0XA3 ;          /* SAVE counters 1 and 2        */


_main ()

BEGIN
        int hrs_, min_ ;
        int temp , ac ;

        ac = getfmt ("%2hi%:%2hi",&hrs_,&min_) ;
        if (ac < 2)
        THEN
                read_time () ;  /* If XX:XX then set time, else read it */
        else
        BEGIN
                set_modes () ;                  /* Mode registers       */
                set_loads (hrs_, min_, 0) ;     /* Load registers       */
                output (CONTROL, &load) ;       /* LCAD command */
                output (CONTROL, &arm) ;        /* ARM command  */
                set_loads (0,0,0) ;     /* Load registers = 0 for next reload */
                read_time () ;                  /* Show if running      */
        END
END
```

Figure 4-9. A Console Driven Clock Written in 'C'

```
set_modes ()

BEGIN
                                /* set master mode reg - recall static init */
        point_to (CTRL_GROUP, MASTER) ;
        output (DATA, &master_reg) ;
                                /* set counter 1 mode - recall dynamic init */
        count[1].mode.output    = OFF_LO_TC ;
        count[1].mode.direction = UP ;
        count[1].mode.base      = BCD ;
        count[1].mode.control   = MODE_DEF ;
        count[1].mode.source    = F5 ;
        count[1].mode.edge      = RISE ;
        count[1].mode.gate      = NO_GATE ;
        point_to (1, MODE_) ;
        output (DATA,&count[1].mode) ;
                                /* set  counter 2 mode  */
        count[2].mode.output    = ACT_HI_TC ;
        count[2].mode.direction = UP ;
        count[2].mode.base      = BCD ;
        count[2].mode.control   = MODE_DEF ;
        count[2].mode.source    = TC_NM1 ;
        count[2].mode.edge      = RISE ;
        count[2].mode.gate      = NO_GATE ;
        point_to (2, MODE_) ;
        output (DATA,&count[2].mode) ;
END


set_loads (hrs, min, secs)

        int hrs, min, secs ;
BEGIN
        count[2].load = (hrs*256) + min ;
        count[1].load = secs * 256 ;
                                /* point and send counter 1 load        */
        point_to (1, LOAD_) ;
        output (DATA,&count[1].load) ;
                                /* point and send counter 2 load        */
        point_to (2, LOAD_) ;
        output (DATA,&count[2].load) ;
END


read_time ()

BEGIN
        unsigned hrs, min, secs, tenths ;

        output (CONTROL, &save) ;       /* Issue the SAVE command       */
        point_to (1, HOLD_) ;
        tenths = in (DATA) ;
        secs = in (DATA) ;              /* Read the Hold 1 register     */
if ((secs == 0) && (tenths == 0))
        THEN                            /* Wups - missed a carry bit    */
                output (CONTROL, &save) ;
        point_to (2, HOLD_) ;           /* Now read correct Hold 2 reg  */
        min = in (DATA) ;
        hrs = in (DATA) ;
        putfmt ("At the bell it was%3hi:%2hi:%2hi\n",hrs,min,secs) ;
END
```

Figure 4-9. A Console Driven Clock Written in 'C' (Cont.)

```
point_to (channel, reg)

BEGIN
        data_type       data_ptr ;

        data_ptr.element = reg ;
        data_ptr.group = channel ;
        data_ptr.cmnd_code = 0 ;
        output (CONTROL,&data_ptr) ;
END


output (port, word)
        unsigned port, *word ;

BEGIN
        unsigned sent ;         /* This output routine has a Trace dump */

        out (port, (*word % 256)) ;
        if (port == DATA)
        THEN
        BEGIN
                out (port, (*word / 256)) ;
                sent = *word ;
        END
        else
                sent = *word % 256 ;

                                /* Delete the next statement to suppress dump
                                   - it shows what you send to the Am9513  */

/* trace off

        putfmt ("Output %2hi%6hi\n",port, sent) :

*/

END


A>
```

Figure 4-9. A Console Driven Clock Written in 'C' (Cont.)

Chapter 5
Event Counting

Chapter 5
Event Counting

EVENT COUNTING

Event counting applications find wide use in industrial control, measurement and telecommunications circuits. These applications are characterized by the need to count transitions on some input signal for a specified duration of time.

In a typical Am9513 event counting interface, the input signal is connected to the counter's source input. The counter is gated on for a controlled period of time either through the hardware gate input or by software command. In some operating modes the gate input can be used to save and reset the counter. The counter will usually be programmed to count up and the Load register will usually be set to 0 to facilitate resetting the counter.

Figure 5-1 shows a simple minimum system for event counting. The counter is operated in Mode D, a repetitive counting mode with no hardware gating. The counter's source input is driven by an external signal which is assumed to provide an active-going edge for each event to be measured. The event accumulation process starts when the counter is loaded with zeroes and armed. After the desired measurement interval, the software can disarm the counter to prevent further count accumulation, save and read the accumulated count, or load the counter with 0 to prepare for a new accumulation cycle. The OUT pin of the counter is connected to the RST 7.5 input of the host Am8085A microprocessor in order to generate an interrupt if the maximum count of 65535 binary or 9999 BCD is exceeded. Note that in Mode D, if the maximum count is exceeded, the counter will continue accumulating counts modulo the maximum count. If the counter is operated in Mode A it will stop after exceeding the maximum count until a new ARM command is issued to it.

One limitation of this event counting technique is the need for software timing loops to control the sampling period. Figure 5-2 eliminates this requirement by utilizing level gating control on the event counter and using another counter to control the gate input and hence the sampling period. The event counter is operated in Mode E and the period counter is operated in Mode D or Mode J for either 50/50 or variable duty cycle output waveforms. A 50/50 waveform (Mode D) can be used to alternate between two event counters, one gated with an active-low signal and the other gated with an active-high signal. A variable duty cycle waveform is most useful when only one event counter is used. In either application, a high signal on the period counter's output interrupts the host microprocessor through RST 5.5. The processor's interrupt routine should read the accumulated event count and reset the event counter to 0. An RST 7.5 interrupt will be issued if the event counter reaches the maximum count. If the maximum count is exceeded, the counter will continue to accumulate modulo the maximum count.

The previous example has eliminated the need for software timing loops but still requires the host processor to reset the event counter after each sampling period. This operation can be performed automatically by the hardware gate input if the event counter is operated in Mode O. Figure 5-3 shows the counter interconnections required for this operating mode. The sampling period counter is operated in Mode J and generates an assymmetric output at fixed intervals. Both the low and high times output by the period counter should be longer thant the execution time for a single pass through the software used to perform steps 1 to 4 below. The event counters's gate is driven by this signal. On each active-going gate edge, the event counter's contents are transferred into the Hold register. On the first active-going source edge after the gate edge, the event counter is reloaded from the Load register. The event counter will start counting on the second source edge after application of the gate edge. Because the gate edge controls saving the accumulated count and resetting the event counter, software intervention is not required. The processor can access the results of the most recent event accumulation cycle by setting the Data Pointer and reading the Hold register contents as described in the following steps.

1. Test the period counter's output by reading the Status register.
2. Set the Data Pointer to point to the counter's Hold register. This step loads the Prefetch Latch with the Hold register's contents.
3. Read the Hold register.
4. Test the period counter's output again.

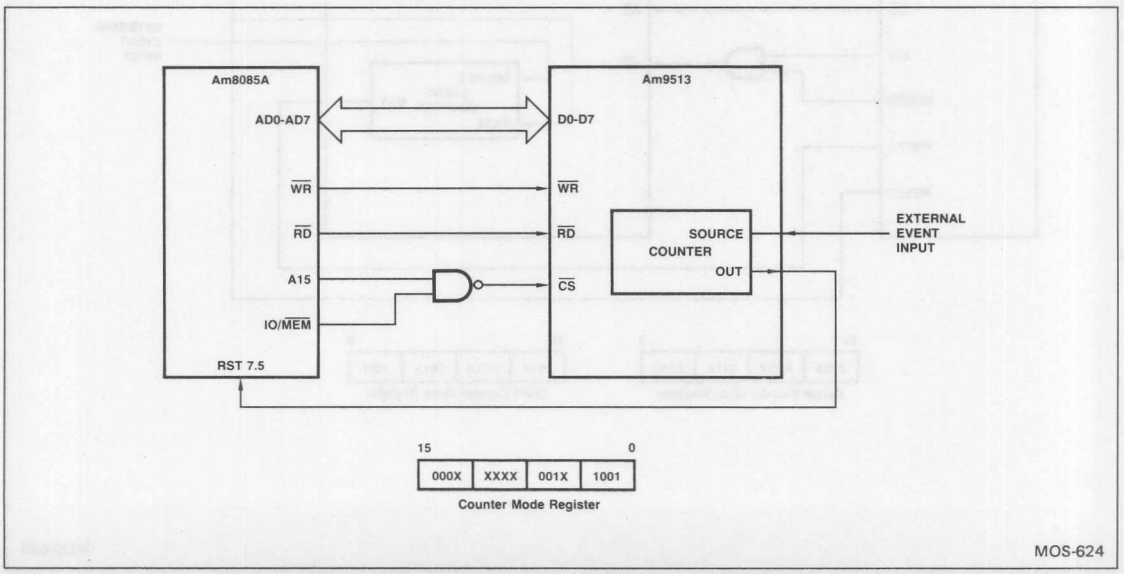


Figure 5-1. Simple One-Counter Event Counting Configuration

If the period counter's output was low in step 1 and high in step 4, an active-going gate edge was applied to the counter during execution of the software. This could potentially cause a faulty read operation for either of two reasons. First, parameters TGVWH and TWHGV in the Am9513 data sheet may have been violated while setting the Data Pointer. See Appendix A for details. Secondly, in an 8-bit bus environment (Master Mode bit MM13 = 0) the gate edge may have split the Hold register values read, with the low byte read reflecting the old Hold register contents and the high byte read reflecting the new updated Hold register contents. Therefore, if the period counter output made a low to high transition between steps 1 and 4, steps 1 through 4 should be repeated. Note that this software must be non-interruptible in order to guarantee that the execution of steps 1 and 4 will be less than half of the period of the period counter's output.

Note that the event counter is reloaded on the first count source edge after application of a gate edge. In most applications, a value of 1 should be reloaded rather than 0, in order to force agreement between the number of count sources applied and the accumulated count. In other words, since the event counter is reloaded by the first source edge, the counter contents after this first pulse should be 1, not 0. Special care must be taken in applications where a possibility exists that no source pulses will be issued to the counter between active-going gate edges. As an illustrative example, consider a counter whose contents are some accumulated value K. When a gate edge is applied to the counter, this value K is transferred to the Hold register. The counter contents at this point are unchanged (i.e., are equal to K). In a normal count cycle, on the next source edge, the counter would be reloaded from the Load register and would start counting events. Consider the situation where no source edges occur before a second active-going gate edge is applied. The counter contents are still K since no reload has occurred, and the Hold register will once again be set to K. A value of K has been saved, even though no source pulses were applied during the sampling period. It can be seen that proper operation of this gate-initiated save/reload operation requires a minimum of 1 active-going source edge between active-going gate edges. Some applications will be able to guarantee this requirement of at least one source edge per sampling cycle by virtue of the signal being measured. Other applications can avoid reading false values by issuing a LOAD command to the counter and then reading the Hold register after at least two sampling periods have elapsed. If the Hold register contents equal the Load register contents, no source pulses are being received by the counter.

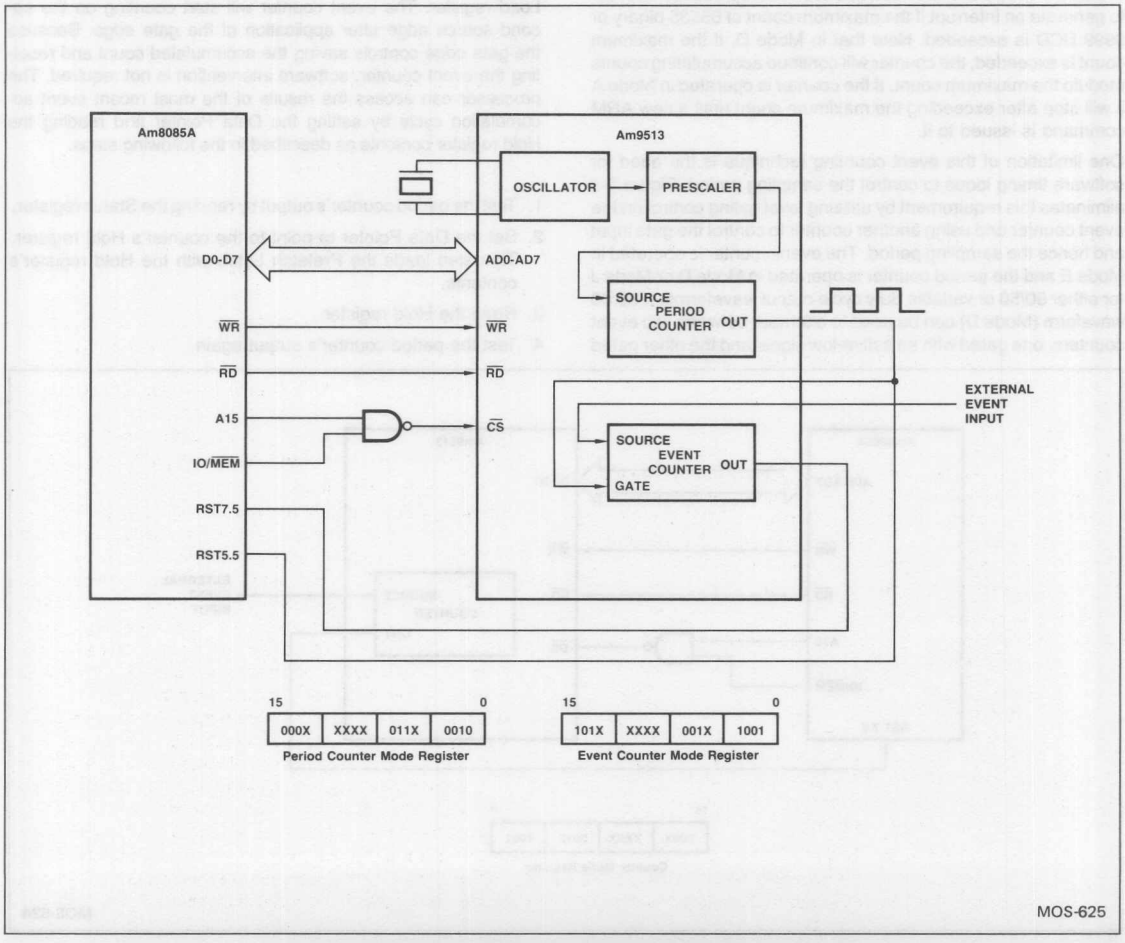


Figure 5-2. Event Counting with Hardware Gating

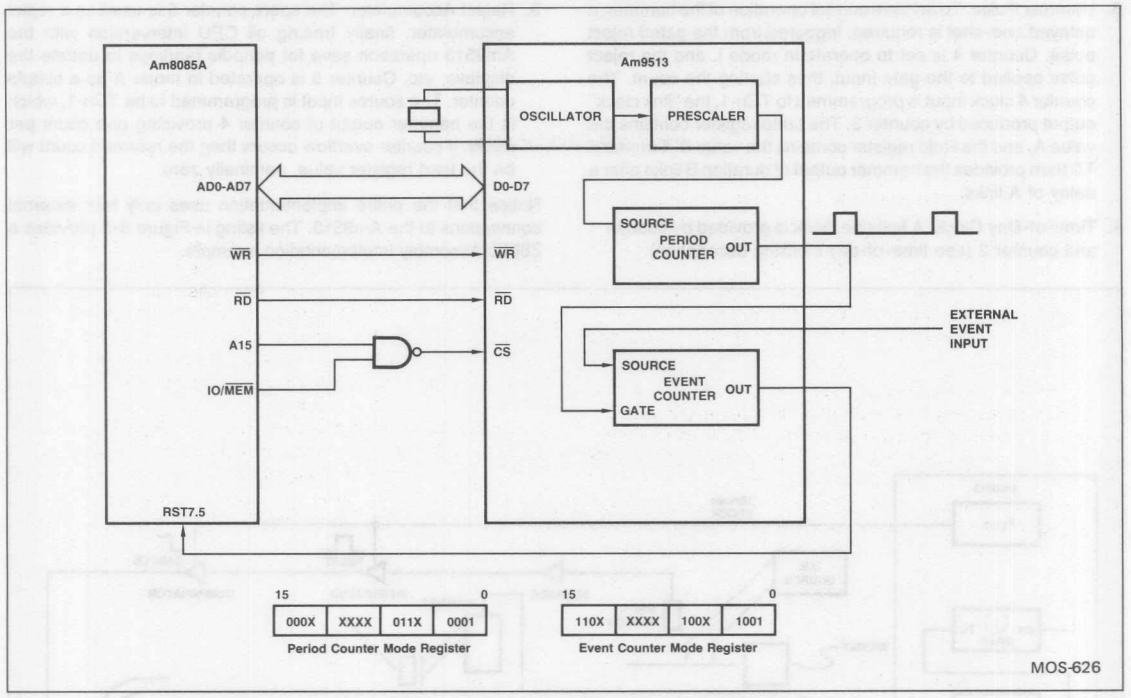


Figure 5-3. Event Counting with Hardware Read and Reset

INDUSTRIAL CONTROLLER – A Z8000 EXAMPLE

A certain manufacturer of widgets has provided the following part specification for the large-volume supply of a microprocessor-based controller system:

"The handling system is constructed around a chain conveyor. Unsorted widgets are loaded onto one end of the conveyor at non-equal intervals, transported along and unloaded at the far end for shipping. Widgets without an identifying mark must be knocked off the conveyor by an existing hammer mechanism, previously manually activated. A small area is available to inspect the widget for the identifying mark a short distance from the reject hammer. The identifying mark fluoresces under the action of ultra-violet illumination of a specific wavelength, otherwise the local area will remain in darkness. The identifying mark, if present, is guaranteed to appear within a defined area."

"The chain conveyor speed is variable within a small known range. Widgets are presented such that a widget does not approach the inspection area until after the preceding widget has left the rejection point. To achieve compatibility with current manual inspection the action of the hammer must be described in terms of chain links. The hammer lies A links beyond the inspection point and must be activated for B links duration to achieve correct rejection. The hammer should not be active outside this period. Sensing of chain conveyor speed must be achieved by non-contact means due to safety regulations."

"Readouts must be provided of current total reject number and the number of rejects per hour. The readings will be reset every 8 hours or so. Widget throughput is around 600 per hour."

Minimizing the component count prompted the system designers to use a single Am9513 device to provide all the counting and timing functions required. The implementation is illustrated in Figure 5-4. The following points discuss the various areas in more detail.

1. Chain Speed Sensor. Unfortunately, due to the chain construction, all available non-contact sensors produce a noisy output in addition to an identifiable edge at the start of each link. A mask pulse is thus required of known maximum duration since the chain speed is known to lie over a small range. Calculations show that synchronization will always be achieved within 20 links, an acceptable power-up time.

   This function is realized by operating counter 3 in mode F. An active edge on the link sensor enables counter 3 to count down from the value stored in the Load register, further gate inputs (link sensor outputs) being ignored until the end of count, or TC. At TC the counter waits for a new gate input to repeat the cycle, in normal operation the commencement of the next link. The clock input to counter 3 is defined to be a suitable internal source, depending on the chain conveyor speed. The TC output of counter 3 cycles at once per link, generating the "link clock" for the system.
2. Widget Mark Sensor. Due to the large area within which the fluorescent mark may appear, the optical sensor produces a fairly slow pulse in the presence of noise and 60Hz and 120Hz interference. For this reason, the sensor output is routed via a high-gain squaring amplifier to an integrator which is reset at 17msec intervals. The integrator cancels most of the interference while providing a clear output of a sensed mark. This output is gated as the integrator reset is applied, generating a single reject pulse if the mark is identified.

   The reset clock is provided by the FOUT output, driven from an appropriate internal source.

3. Hammer Pulse. To achieve correct operation of the hammer, a delayed one-shot is required, triggered from the gated reject pulse. Counter 4 is set to operate in mode L and the reject pulse applied to the gate input, thus starting the count. The counter 4 clock input is programmed to TCn-1, the "link clock" output produced by counter 3. The Load register contains the value A, and the Hold register contains the value B. Counter 4 TC then provides the hammer output of duration B links after a delay of A links.
4. Time-of-Day Clock. A suitable clock is provided by counter 1 and counter 2 (see time-of-day clocking description).
5. Reject Accumulator. The spare counter 5 is used as a reject accumulator, finally freeing all CPU intervention with the Am9513 operation save for periodic readings to update the displays, etc. Counter 5 is operated in mode A as a simple counter. The source input is programmed to be TCn-1, which is the hammer output of counter 4 providing one count per reject. If counter overflow occurs then the returned count will be the load register value, nominally zero.

Notice that the entire implementation uses only four external connections to the Am9513. The listing in Figure 5-5 provides a Z8000 Assembly implementation example.

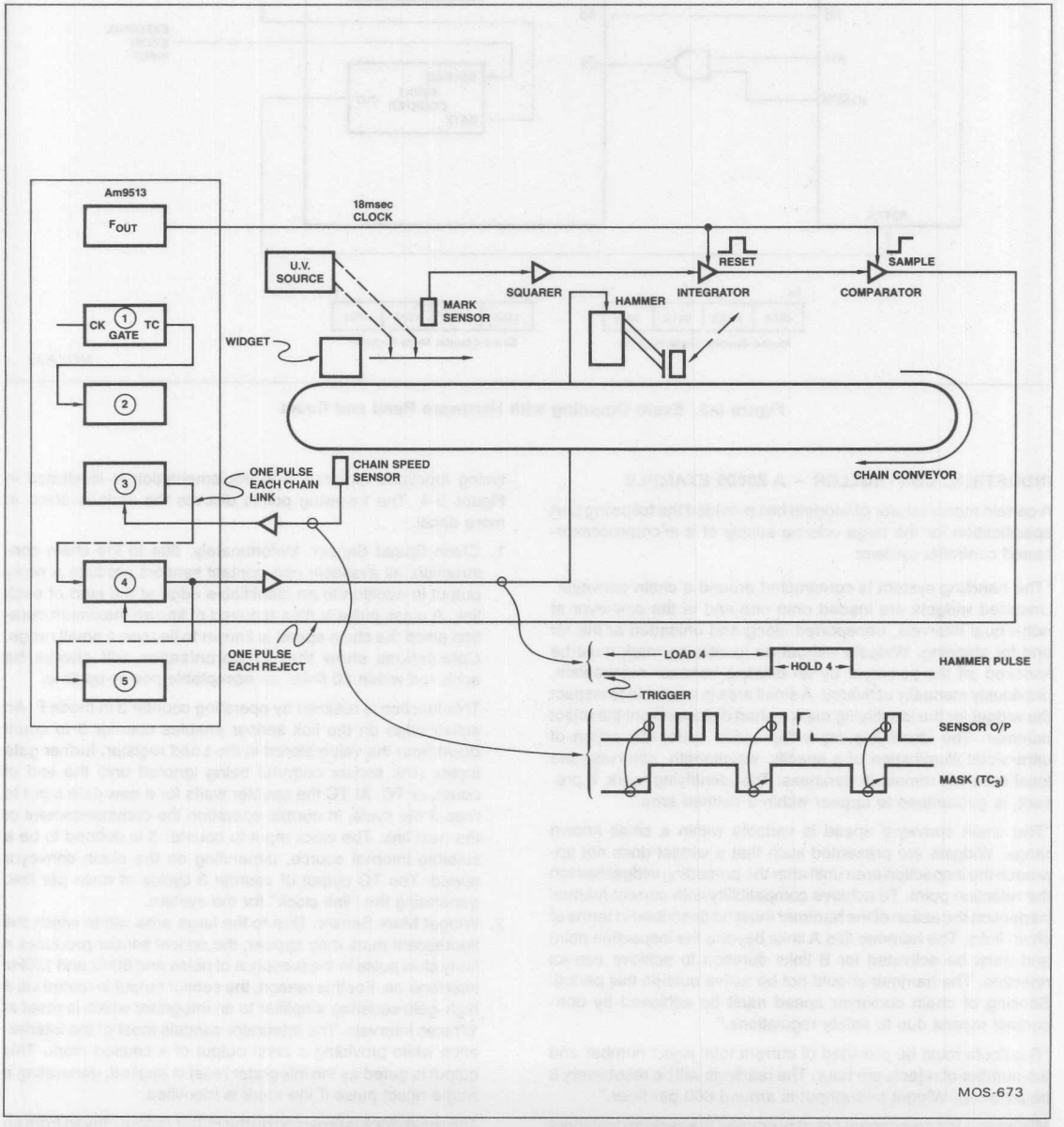


Figure 5-4. Am9513 Industrial Controller

```
MODULE "WIDGET";

%       Coded for AmZ8000

%       This file provides a short industrial controller example using the
% Am9513 System Timing Controller as a multi-purpose timing and counting
% element.

%       Warning: Due to the nature of this file it has not been fully tested.

%       This file is part of the Am9513 Software Applications Manual,
% is not copyright and you can store it on what you like.


CONST
A_DATA      =    0FFD8H          , % Am9513 data port
A_CTRL      =    0FFDAH          , % Am9513 control port
CHAIN_MASK  =    5000            , % Mask pulse for chain sensor noise
A_LINKS     =    42              , % Activate hammer a_links after inspection
B_LINKS     =    7               ; %  ... for b_links duration


INCLUDE 'B:EQUSZ.ZSC'            ; % Listed in Appendix 7
INCLUDE 'B:Z8000MAC.ZSC'         ; % Listed in Appendix 5

ORIGIN  9000H ;


WIDGET:
        % Reset and initialise the Am9513.

        RESET                    ; % Reset, Load all, set 16 bit bus, set data ptr

        MASTER TOD_100HZ,DISABLE,DISABLE,F5,10,ON,BUS_16,ON,BCD ;

        CALR SETTIME             ; % Get the time of day clock set up

        MODE_REG 3,ACT_HI_TC,DOWN,BCD,MODE_DEF,F4,RISE,HE_GATE_N ;
        LOAD_REG 3,CHAIN_MASK    ; % Chain speed sensor

        MODE_REG 4,TC_TOGGLE,DOWN,BCD,MODE_JKL,TC_NM1,RISE,HE_GATE_N ;
        LOAD_REG 4,A_LINKS       ;
        HOLD_REG 4,B_LINKS       ; % Reject hammer output

        MODE_REG 5,OFF_OC_TC,UP,BINARY,MODE_ABC,TC_NM1,RISE,NO_GATE ;
        LOAD_REG 5,0             ; % Widget reject counter

        CLEAR    3               ; % TC outputs must start low for sync
        CLEAR    4               ;
        ARM      1,2,3,4,5       ; % Time and action starts now
        RET                      ;
```

Figure 5-5. AmZ8000 Assembly Listing for Controller

```
SETTIME:
        % Set up channels 1 and 2 to time of day mode

        MODE_REG 1,OFF_OC_TC,UP,BCD,MODE_DEF,F5,PISE,NO_GATE;

        MODE_REG 2,ACT_HI_TC,UP,BCD,MODE_DEF,TC_NM1,RISE,NO_GATE;

        LOAD_REG 1,Ø            ; % Ø here is a constant
        LOAD_REG 2,Ø            ;

        LOAD     1,2            ; % Enable TOD counting

        LOAD_REG 1,TIME(2)      ; % Current time value - notice no I flag reqd.
        LOAD_REG 2,TIME         ;

        LOAD     1,2            ; % Set the current time

        LOAD_REG 1,Ø            ;
        LOAD_REG 2,Ø            ; % Recall repeat reload values needed

        RET                     ;


READ_IT:
        % Routine to get current system status.
        %
        % Exit: R1    holds time XX:XX (hrs:min) in 4 x BCD format
        %       R2    holds current total reject #

        SAVE    1,2,5           ; % Fix the data
        POINT   1,HOLD_         ;
        IN      RØ,A_DATA       ; % Read secs, tenths
        TEST    RØ              ;
        JR      NZ,READ_IT1     ; % Read was ok

        SAVE    2               ; % We might have missed a carry bit
READ_IT1:
        POINT   2,HOLD_         ;
        IN      R1,A_DATA       ; % Read hrs, mins
        POINT   5,HOLD_
        IN      R2,A_DATA       ; % Read current reject total
        RET                     ;

                        % Required start time hrs, min, secs, tenths

TIME:   BYTE:   Ø9H,15H,Ø,Ø      ;  % About the time we get to work (9:15am)

END.


A>
```

Figure 5-5. AmZ8000 Assembly Listing for Controller (Cont.)

Chapter 6
Frequency and Baud Rate Generation

Chapter 6
Frequency and Baud Rate Generation

FREQUENCY GENERATION

The Am9513 is capable of synthesizing frequencies with a high degree of precision. For square wave outputs, the counter's Mode register should be set for a TC Toggled output. In this configuration, the output will generate a complete cycle for every two TCs the counter generates. Thus, if the spacing between TCs is controlled by the Load register, the TC Toggled output period (in number of source pulses) will be twice the Load register contents.

A 50% duty cycle output can be generated by operating the counter in Mode D. In this mode, the time between TCs is controlled by the Load register contents. The Hold register is not used in controlling the counter. Figure 6-1 shows an Am8080A/8085A routine for configuring Counter 3 as a baud rate generator.

Some applications require square wave outputs with variable duty cycles. This can be achieved by operating the counter in Mode J. In this mode the reload sources alternate between the

```
0100                00010           ORG 100H
                    00020           ;
                    00030           ;EQU'S FOR AM9513 PORTS
                    00040           ;
0012 =              00050   CMDPRT  EQU     12H     ;COMMAND PORT
0010 =              00060   DATAPRT EQU     10H     ;DATA PORT
                    00070           ;
                    00080           ;
                    00090           ;
                    00100           ;THIS SUBROUTINE CONFIGURES COUNTER 3 ON
                    00110           ;THE AM9513 AS A BAUD RATE GENERATOR. THE
                    00120           ;INTERNAL OSCILLATOR FREQUENCY IS DIVIDED
                    00130           ;DOWN BY TWICE THE CONTENTS OF VARIABLE
                    00135           ;'FACTOR' AND A SQUARE WAVE OUTPUT IS
                    00137           ;GENERATED.
                    00140           ;
                    00150   BAUD:
                    00160           ;SET COUNTER MODE AND LOAD REGISTERS.
                    00170           ;
0100 3EE0           00171           MVI     A,0E0H  ;ENABLE DATA POINTER
0102 D312           00172           OUT     CMDPRT  ;SEQUENCING.
0104 3E03           00180           MVI     A,03H   ;SET DATA POINTER TO
0106 D312           00190           OUT     CMDPRT  ;COUNTER 3 MODE REG.
0108 3E22           00200           MVI     A,22H   ;LOWER BYTE
010A D310           00210           OUT     DATAPRT
010C 3E0B           00220           MVI     A,0BH   ;UPPER BYTE
010E D310           00230           OUT     DATAPRT
                    00240           ;
                    00250           ;SET COUNTER 3 LOAD REG.
                    00260           ;
0110 3A2301         00270           LDA     FACTOR  ;LOWER BYTE
0113 D310           00280           OUT     DATAPRT
0115 3A2401         00290           LDA     FACTOR+1        ;UPPER BYTE
0118 D310           00300           OUT     DATAPRT
                    00310           ;
                    00320           ;LOAD AND ARM COUNTER
                    00330           ;
011A 3E44           00340           MVI     A,44H   ;LOAD COUNTER 3 COMMAND
011C D312           00350           OUT     CMDPRT
011E 3E24           00360           MVI     A,24H   ;ARM COUNTER 3 COMMAND
0120 D312           00370           OUT     CMDPRT
0122 C9             00380           RET
                    00390           ;
                    00400           ;INTERNAL OSCILLATOR FREQUENCY WILL BE
                    00410           ;DIVIDED BY TWICE CONTENTS OF
                    00420           ;VARIABLE 'FACTOR'.
                    00422           ;
0123 1000           00430   FACTOR: DW      16      ;DIVIDE BY 32
                    00432           ;
                    00440           ;A CRYSTAL OF FREQUENCY 4.915200
                    00450           ;MHZ. WHEN DIVIDED BY 32 WILL YIELD
                    00460           ;AN OUTPUT FREQUENCY OF 9600 X 16 =
                    00470           ;153600 HZ. FOR 9600 BAUD SERIAL
                    00480           ;COMMUNICATIONS.
                    00490           ;
0125                00500           END
```

Figure 6-1. Baud Rate Generation with the Am9513 in an Am8080A/8085A System

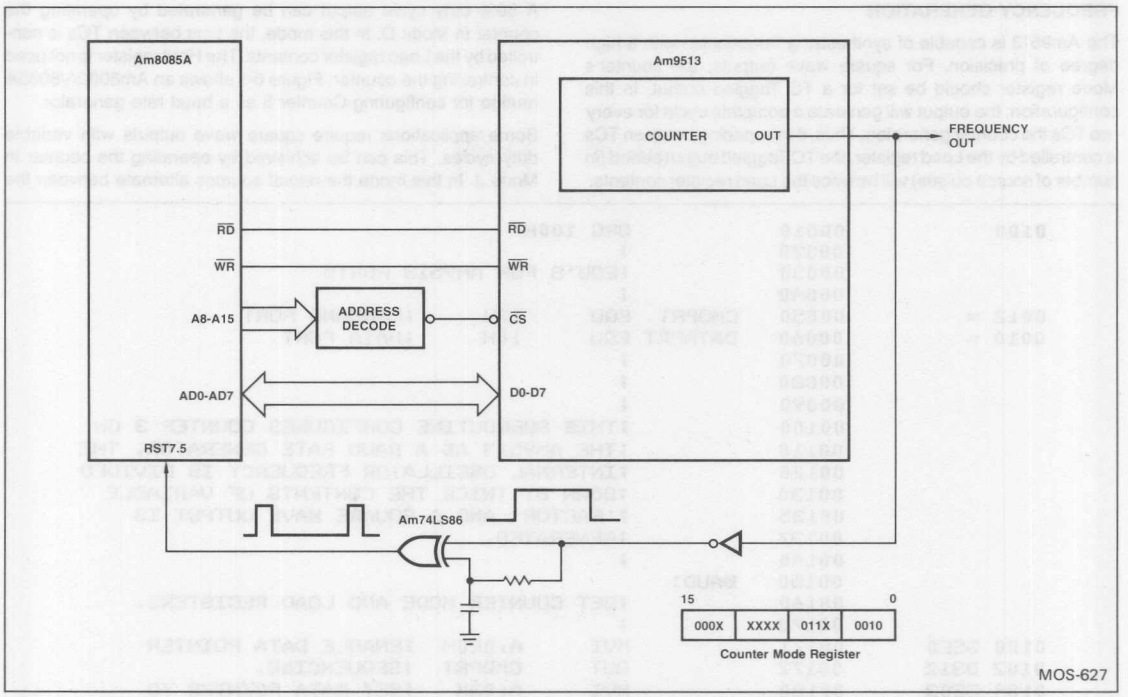


Figure 6-2. Updating Load and Hold Register Values in Real-Time

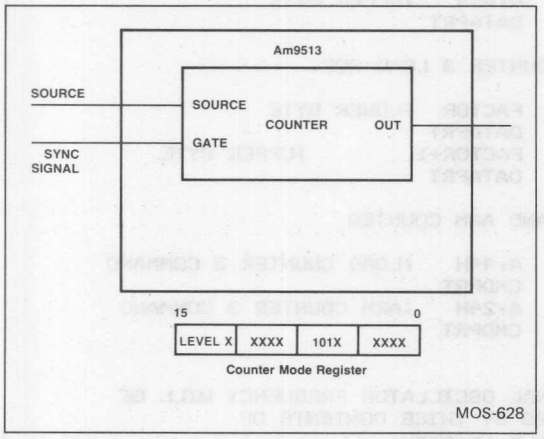


Figure 6-3. External Synchronization of Frequency Counters: Method 1

Load and Hold registers. This allows one polarity of the output to be controlled by the Load register contents and the other output polarity to be controlled by the Hold register. Fine resolution of the output duty cycle can be achieved by adjusting the relative values of the two registers.

In some applications it is necessary to adjust the output frequency in real time. Figure 6-2 shows a circuit which generates an interrupt on each edge of the output waveform. The host microprocessor's interrupt routine would update the Load and/or Hold register contents to adjust the period of the next output cycle. Updating the registers by means of an interrupt routine ensures the register contents are stable when the counter reaches TC and reloads from the register.

The Am9513 has provisions for synchronizing a counter's output frequency to an external signal. To accomplish synchronization, the counter should be operated in Mode Q, as shown in Figure 6-3. For normal counter operation the gate input is held active. While the gate is active, the counter will count to TC repeatedly. When it is necessary to synchronize the counter to an external signal, the gate should be driven inactive, which will stop counting, and then driven active again. On the active-going gate edge, the current counter contents will be saved in the Hold register. This value may be used by the microprocessor as an indication of how out-of-sync the counter was. On the first source edge after the gate edge, the counter will be reloaded from the Load register. Counting will start (assuming the gate remains active) on the second source edge after the gate edge. It can be seen that the gate edge effectively resets the counter, synchronizing it to the gate signal.

Synchronization of the counter to an external event can also be accomplished by issuing a LOAD command to the counter. This will cause the Load or Hold register contents to be transferred into the counter. The selection of whether the Load or Hold register is used as a reload source depends on the counter's operating mode. Note that data sheet parameters TEHWH, TWHEH, TGVWH and TWHGV must be met to ensure successful synchronization. See Appendix A for further details.

Applications which need to generate Frequency-Shift Keying outputs can achieve this function by operating the counter in Mode V. This mode uses the level of the gate input to select the

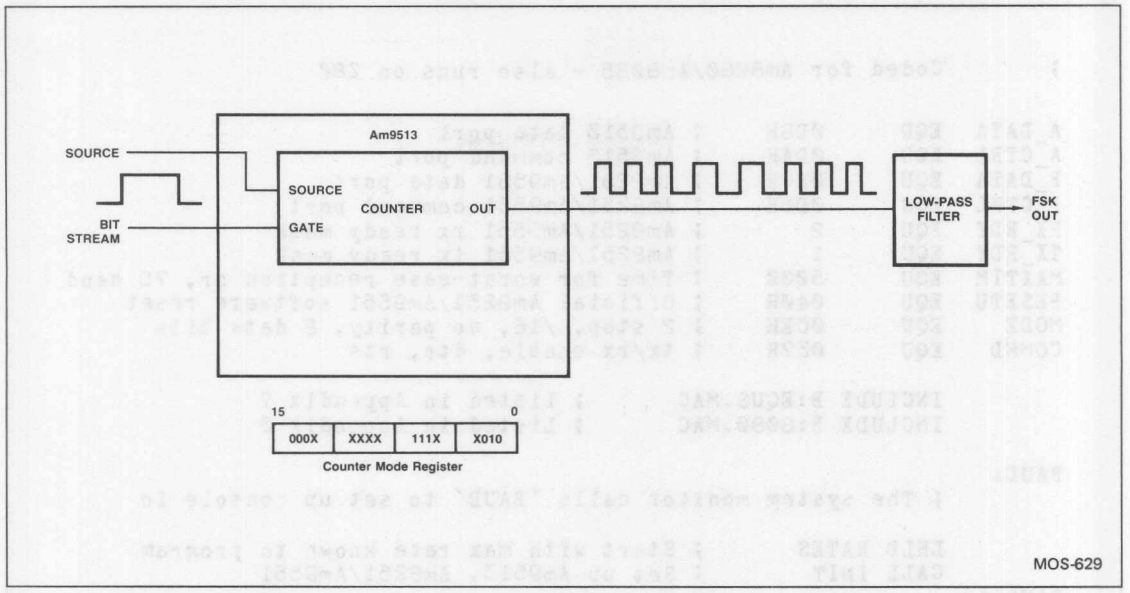


Figure 6-4. Frequency Shift Keying

counter reload source for each TC. For Frequency-Shift Keying (FSK) applications, the serial bit stream should be input to the counter's gate and the counter output periods (as set by the Load and Hold register contents) should be selected to be less than the shortest high or low time on the incoming bit stream. The FSK output can be shaped by a low-pass filter to remove high order harmonics. The output can then be transmitted over a telecommunications link or stored on an audio tape recorder for a low-cost data acquisition system. Demodulation of the FSK signal can be performed using standard techniques.

AUTO BAUD RATE GENERATOR

Many systems require an asynchronous data link that is typically implemented using an Am9513 System Timing Controller and Am8251/Am9551 USART. Since these systems are often shipped without accompanying CRT consoles, they may be set at an arbitrary baud rate to cater for common terminals, usually by means of a buffered bank of dip switches and associated address decode logic. A system capable of automatically sensing the attached terminal baud rate at power-up would save not only switch adjustment by the end user (frequently after dismantling the processor) but also reduce component count, board space and assembly effort at the expense of a few lines of code. Such an example usage of the Am9513 is presented here, using about 160 bytes of code (Am8080/Am8085).

The principle of the baud rate determination algorithm is unrefined but effective. The Am9513 baud rate generator is set to provide the maximum rate anticipated, perhaps 19,200 baud. The system user first enters a carriage return, thus allowing the terminal rate to be measured by counting the pseudo characters received by the system within a short time frame. Some of these characters will be flagged with framing errors by the Am8251/Am9551, although these errors may be safely ignored for our purposes.

The pseudo characters seen by the system, for a given terminal baud rate and asynchronous character format, may be accurately predicted in advance. For example, a 4800 baud terminal will produce the four-character sequence ⟨78⟩⟨7E⟩⟨00⟩⟨7E⟩. However, if the number of pseudo characters received is used as the primary parameter the measurement becomes much more tolerant to skew and variation in the number of bits per word. In practice, many other characters in addition to a carriage return provide correct measurement. The "capture range" of characters entered may be increased by reducing the number of allowable baud rates.

Am8080/Am8085 SAMPLE LISTING

The Am8080/Am8085 code sequence shown in Figure 6-5 initializes the Am8251/Am9551 and the Am9513 to provide an asynchronous data link at 19,200 baud. The received characters are decoded and the new baud rate time constant calculated by using an indexed sequential search. To reduce code, the entire initialization of the Am9513 and Am8251/Am9551 is then recalled using the calculated baud rate time constant. Notice that this practice avoids problems with asynchronous changes of the Am9513 load register when counting in close proximity to TC.

AmZ8000 SAMPLE LISTING

The Z8000 code sequence shown in Figure 6-6 performs the same action as the previous example with the additional facility of changing the Am8251/Am9551 Mode control word for differing baud rates. For example, at 9600 baud frequently 10 bits per byte are utilized compared to 11 bits per byte at 110 baud.

For both examples, successful decoding of the baud rate is indicated by the appearance of the prompt (")") on the users terminal.

Additional baud rates may be allowed by entering the appropriate data in the two data tables (TABLE and RATES). Reducing the number of baud rates available will allow a greater number of characters to give correct operation, although the program is designed to receive a carriage return. As listed, the 11 standard baud rates between 75 and 19,200 baud are allowed.

```
;       Coded for Am8080/Am8085 - also runs on Z80

A_DATA  EQU     0D8H    ; Am9513 data port
A_CTRL  EQU     0DAH    ; Am9513 command port
B_DATA  EQU     0DCH    ; Am8251/Am9551 data port
B_CTRL  EQU     0DDH    ; Am8251/Am9551 command port
RX_RDY  EQU     2       ; Am8251/Am9551 rx ready mask
TX_RDY  EQU     1       ; Am8251/Am9551 tx ready mask
MAXTIM  EQU     5000    ; Time for worst-case reception or, 75 baud
RESETU  EQU     040H    ; Official Am8251/Am9551 software reset
MODE    EQU     0CEH    ; 2 stop, /16, no parity, 8 data bits
COMND   EQU     037H    ; tx/rx enable, dtr, rts

        INCLUDE B:EQUS.MAC      ; Listed in Appendix 7
        INCLUDE B:8080.MAC      ; Listed in Appendix 3

BAUD:
        ; The system monitor calls 'BAUD' to set up console io

        LHLD RATES      ; Start with max rate known to program
        CALL INIT       ; Set up Am9513, Am8251/Am9551
BAUD_1:
        IN   B_CTRL     ; Read console status
        ANI  RX_RDY
        JZ   BAUD_1     ; Wait for 1st character

        MVI  E,0        ; Zero character counter
        LXI  B,MAXTIM   ; Initialise timer
BAUD_2:
        DCX  B          ; Move and test timer status
        MOV  A,B
        CRA  C
        JZ   BAUD_3     ; Timeout, go decode data

        IN   B_CTRL     ; Read console status
        ANI  RX_RDY
        JZ   BAUD_2     ; Wait for a character

        IN   B_DATA     ; Discard pseudo character
        INR  E          ; Adjust character counter
        JMP  BAUD_2     ; Repeat
BAUD_3:
        ; Perform serial index search for reqd baud rate const
        ;
        ; Entry:  E holds # pseudo characters received

        MOV  A,E
        LXI  H,TABLE-1  ; Byte pointer, decode table
        LXI  D,RATES-2  ; Word pointer, rate constant table
BAUD_4:
        INX  D          ; Move the pointers
        INX  D
        INX  H
        CMP  M          ; Check the decode entry
        JNC  BAUD_4     ; Not found yet
```

Figure 6-5. Am8080/8085 Auto Baud Rate Generator

```
        ; DE points to the required time constant


        LDAX D          ; Put time constant in HL
        MOV  L,A
        INX  D
        LDAX D
        MOV  H,A
        CALL INIT       ; Re-program both Am9513 and Am8251/Am9551
BAUD_5:
        ; Print a log-on prom . to indicate success

        IN   B_CTRL     ; Read console status
        ANI  TX_RDY
        JZ   BAUD_5     ; Wait till ready - good practice

        MVI  A,'>'      ; Send the prompt
        OUT  B_DATA
        RET

INIT:
        ; Reset and set Am9513 channel 1 to baud rate generator.
        ; Mode: TC toggle, Mode D, count bin/down, SRC 1
        ; Note: For Am9513 crystal use F1. SRC 1 used here for testing.
        ; Reset and set Am8251/Am9551 to usual async uart
        ;
        ; Entry: HL contains baud rate time constant

        RESET
        MODE_REG 1,TC_TOGGLE,DOWN,BINARY,MODE_DEF,SRC_1,RISE,NO_GATE
        POINT    1,LOAD_
        MOV  A,L        ; Set up counter 1 LOAD register
        OUT  A_DATA
        MOV  A,H
        OUT  A_DATA     ; That was the upper byte
        LOAD 1          ; Transfer the value
        ARM  1          ; Turn on the baud rate generator

        ; Safe now to set up the Am8251/Am9551 uart

        MVI  A,Ø        ; Ensure no sync problems
        OUT  B_CTRL
        OUT  B_CTRL
        OUT  B_CTRL
        MVI  A,RESETU   ; Am8251/Am9551 software reset
        OUT  B_CTRL
        MVI  A,MODE     ; Set uart mode
        OUT  B_CTRL
        MVI  A,COMND    ; Set uart command register
        OUT  B_CTRL
        IN   B_DATA     ; Clear any rubbish around
        RET
```

Figure 6-5. Am8080/8085 Auto Baud Rate Generator (Cont.)

```
TABLE:
        ; Conversion table - used to calculate an index from the
        ; number of pseudo characters received by the system.
        ; Entries may be adjusted for convenience of use.

        DB      2       ; 19200 baud, nominally 1 char: ie match if < 2
        DB      4       ; 9600  nom 2
        DB      5       ; 4800  nom 4
        DB      7       ; 2400  nom 5
        DB      9       ; 1800  nom 8
        DB      13      ; 1200  nom 9
        DB      23      ; 600   nom 18
        DB      45      ; 300   nom 34
        DB      75      ; 150   nom 68
        DB      108     ; 110   nom 93
        DB      255     ; 75    nom >130

RATES:
        ; Baud rate constant conversion table. The indexed entry
        ; corresponds to the value required for the Am9513 LOAD
        ; register if a 2.4576MHz crystal is used. Notice the one to
        ; one correspondence between the entries in RATES and TABLE.
        ;
        ; Calculate new rates (or different crystals) by:
        ;
        ;       entry := ((crystal/32)/baud_rate)
        ;
        ; All values in Hz. The factor 32 comes from /16 mode in
        ; the Am8251/Am9551 uart and /2 from the Am9513 TC toggle mode.

        DW      4       ; 19200 baud
        DW      8       ; 9600
        DW      16      ; 4800
        DW      32      ; 2400
        DW      43      ; 1800
        DW      64      ; 1200
        DW      128     ; 600
        DW      256     ; 300
        DW      512     ; 150
        DW      698     ; 110
        DW      1024    ; 75

END BAUD

A>
```

Figure 6-5. Am8080/8085 Auto Baud Rate Generator (Cont.)

AmZ8000 SAMPLE LISTING

The Z8000 code sequence shown in Figure 6-6 performs the same action as the previous example with the additional facility of changing the Am8251/Am9551 Mode control word for differing baud rates. For example, at 9600 baud frequently 10 bits per byte are utilized compared to 11 bits per byte at 110 baud.

For both examples, successful decoding of the baud rate is indicated by the appearance of the prompt (")") on the users terminal.

Additional baud rates may be catered for by entering the appropriate data in the two data tables (TABLE and RATES). Reducing the number of baud rates available will allow a greater number of characters to give correct operation, although the program is designed to receive a carriage return. As listed, the 11 standard baud rates between 75 and 19,200 baud are catered for.

```
%       This file allows auto baud-rate sensing of an attached terminal
% by looking at the data returned from a user entered carriage return.
% It assumes the use of an Am9513, Am8251/Am9551 combination.

CONST
A_DATA  =       0FFD8H, % Am9513 data port
A_CTRL  =       0FFDAH, % Am9513 command port
B_DATA  =       0FFDCH, % Am8251/Am9551 data port
B_CTRL  =       0FFDDH, % Am8251/Am9551 command port
RX_RDY  =       1,      % Am8251/Am9551 rx ready bit
TX_RDY  =       0,      % Am8251/Am9551 tx ready bit
MAXTIM  =       5000,   % Time for worst-case reception cr, 75 baud
RESETU  =       040H,   % Cfficial Am8251/Am9551 software reset
MODE_1  =       0CEH,   % 2 stop, /16, no parity, 8 data bits
MODE_2  =       0DEH,   % 2 stop, /16, odd parity, 8 data bits
COMND   =       037H,   % tx/rx enable, dtr, rts
INDEX   =       R1,     % Common index for TABLE and RATES
TIMER   =       R2,     % Timer for worst case reception
COUNT   =       RL3;    % Input psuedo character counter

INCLUDE 'B:EQUSZ.ZSC'     ; % Listed in Appendix 7
INCLUDE 'B:Z8000MAC.ZSC'  ; % Listed in Appendix 5

BAUD:
        % The system monitor calls 'BAUD' to set up console io

        CLR  INDEX        ; % Start with max rate known to program
        CALR INIT         ; % Set up Am9513, Am8251/Am9551
BAUD_1:
        IN   R0,B_CTRL    ; % Read console status
        BIT  R0,RX_RDY    ;
        JR   ZR,BAUD_1    ; % Wait for 1st character

        CLRB CCUNT        ; % Zero character counter
        LD   TIMER,MAXTIM ; % Initialise timer
BAUD_2:
        DEC  TIMER,1      ; % Move and test timer status
        JR   ZR,BAUD_3    ; % Timeout, go decode data

        IN   R0,B_CTRL    ; % Read console status
        BIT  R0,RX_RDY    ;
        JR   ZR,BAUD_2    ; % Wait for a character

        IN   R0,B_DATA    ; % Discard pseudo character
        INCB COUNT,1      ; % Adjust character counter
        JR   BAUD_2       ; % Repeat
BAUD_3:
          ; % Perform serial index search for reqd baud rate const
          ; % Entry:  COUNT holds # pseudo characters received

        LD   INDEX,-2     ; % Word index
BAUD_4:
        INC  INDEX,2      ; % Move the index
        CPB  COUNT,TABLE(INDEX) ; % Check the decode entry
        JR   NC,BAUD_4    ; % Not found yet
```

Figure 6-6. Z8000 Auto Baud Rate Generator

```
        % INDEX is offset for both baud rate constant and Am8251/Am9551
        % command byte

        CALR INIT           ; % Re-program both Am9513 and Am8251/Am9551
BAUD_5:
        % Print a log-on prompt to indicate success

        IN   RØ,B_CTRL      ; % Read console status
        BIT  RØ,TX_RDY      ;
        JR   ZR,BAUD_5      ; % Wait till ready - good practice

        LDB  RLØ,'>'        ; % Send the prompt
        OUT  B_DATA,RØ      ;
        RET                 ;


INIT:
        % Reset and set Am9513 channel 1 to baud rate generator.
        % Mode: TC toggle, Mode D, count bin/down, SRC 1
        % Note: For Am9513 crystal use F1. SRC 1 used here for testing.
        % Reset and set Am8251/Am9551 to usual async uart
        %
        % Entry: INDEX contains baud rate time constant index


        RESET               ; % Not neccessary if the Am9513 already set
        MODE_REG  1,TC_TOGGLE,DOWN,BINARY,MODE_DEF,SRC_1,RISE,NO_GATE;
        POINT 1,LOAD        ;
        LD   RØ,RATES(INDEX) ; % Get rate constant
        OUT  A_DATA,RØ      ;
        LOAD 1              ;
        ARM  1              ;

        % Safe now to set up the Am8251/Am9551 uart

        CLR  RØ             ; % Ensure no sync problems
        OUT  B_CTRL,RØ      ;
        OUT  B_CTRL,RØ      ;
        OUT  B_CTRL,RØ      ;
        LD   RØ,RESETU      ; % Am8251/Am9551 software reset
        OUT  B_CTRL,RØ      ;
        LD   RØ,TABLE(INDEX) ; % Set uart mode
        OUT  B_CTRL,RØ      ;
        LD   RØ,COMND     ; % Set uart command register
        OUT  B_CTRL,RØ      ;
        IN   RØ,B_DATA      ; % Clear any rubbish around
        RET                 ;
```

Figure 6-6. Z8000 Auto Baud Rate Generator (Cont.)

```
TABLE:
        % Conversion table - used to calculate an index from the
        % number of pseudo characters received by the system.
        % Notice that a different mode byte may be defined for
        % each separate baud rate - 110 baud should use 11 bits.


        BYTE: 2,   MODE_1       ; % 19200 baud, nominally 1 char.
        BYTE: 4,   MODE_1       ; % 9600  nom 2              Match < 2
        BYTE: 5,   MODE_1       ; % 4800  nom 4
        BYTE: 7,   MODE_1       ; % 2400  nom 5
        BYTE: 9,   MODE_1       ; % 1800  nom 8
        BYTE: 13,  MODE_1       ; % 1200  nom 9
        BYTE: 23,  MODE_1       ; % 600   nom 18
        BYTE: 45,  MODE_1       ; % 300   nom 34
        BYTE: 75,  MODE_2       ; % 150   nom 68
        BYTE: 108, MODE_2       ; % 110   nom 93
        BYTE: 255, MODE_2       ; % 75    nom >130


RATES:
        % Baud rate constant conversion table. The indexed entry
        % corresponds to the value required for the Am9513 LOAD
        % register if a 2.4576MHz crystal is used. Notice the one to
        % one correspondence between the entries in RATES and TABLE.
        %
        % Calculate new rates (or different crystals) by:
        %
        %       entry := ((crystal/32)/baud_rate)
        %
        % All values in Hz. The factor 32 comes from /16 mode in
        % the Am8251/Am9551 uart and /2 from the Am9513 TC toggle mode.


        WORD:    4      ; % 19200 baud
        WORD:    8      ; % 9600
        WORD:    16     ; % 4800
        WORD:    32     ; % 2400
        WORD:    43     ; % 1800
        WORD:    64     ; % 1200
        WORD:    128    ; % 600
        WORD:    256    ; % 300
        WORD:    512    ; % 150
        WORD:    698    ; % 110
        WORD:    1024   ; % 75

END.


A>
```

Figure 6-6. Z8000 Auto Baud Rate Generator (Cont.)

Chapter 7
One-Shot Applications

Chapter 7
One-Shot Applications

ONE-SHOT OPERATING MODES

The Am9513 is capable of providing a variety of retriggerable and non-retriggerable one-shot functions, usually with no external logic. One-shot timing functions may be triggered in one of three manners: by a software ARM command, by an active-going gate edge, or by an ARM command followed by a gate edge.

Triggering by a software ARM command, shown in Figure 7-1a, is provided by Mode A. This triggering mode is particularly useful when the one-shot's firing is to be controlled by the microprocessor, independent of any external signals.

Triggering by a hardware gate edge, shown in Figure 7-1b, is provided by Mode F for non-retriggerable one-shot operation, and by Mode R, for retriggerable one-shot operation. This trigger type is useful when the one-shot is to be repetitively fired by an external signal, independent of the host microprocessor. When using this triggering option, the next count cycle can be triggered after the counter reaches TC. In fact, if the TC output is fed back to the counter's gate, as shown in Figure 7-2, the counter will essentially free-run after being started by an external gate pulse. The counter can trigger off either edge of the TC pulse in this configuration. Figure 7-2 shows the TC output being fed back through

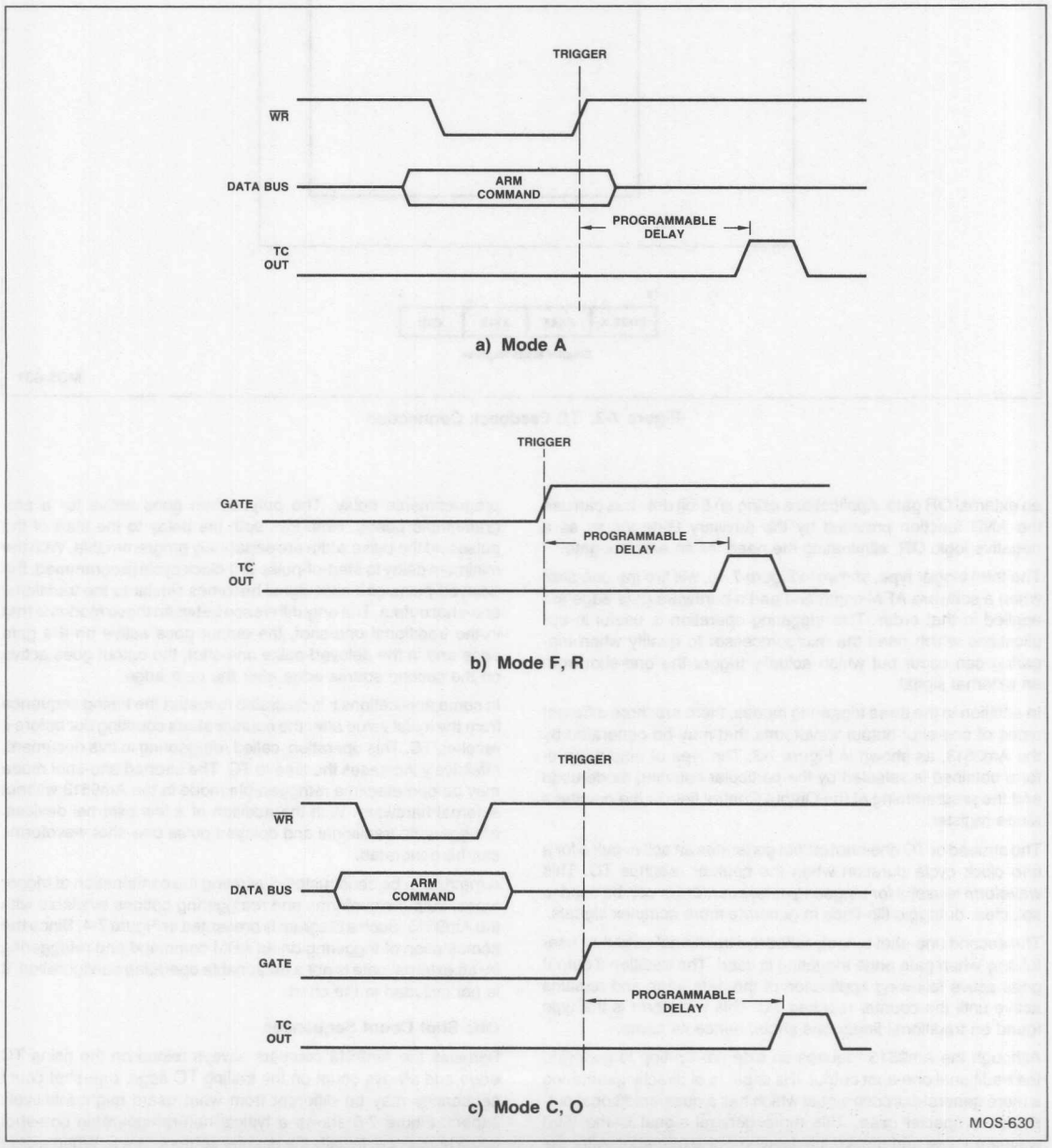


Figure 7-1. One-Shot Trigger Options

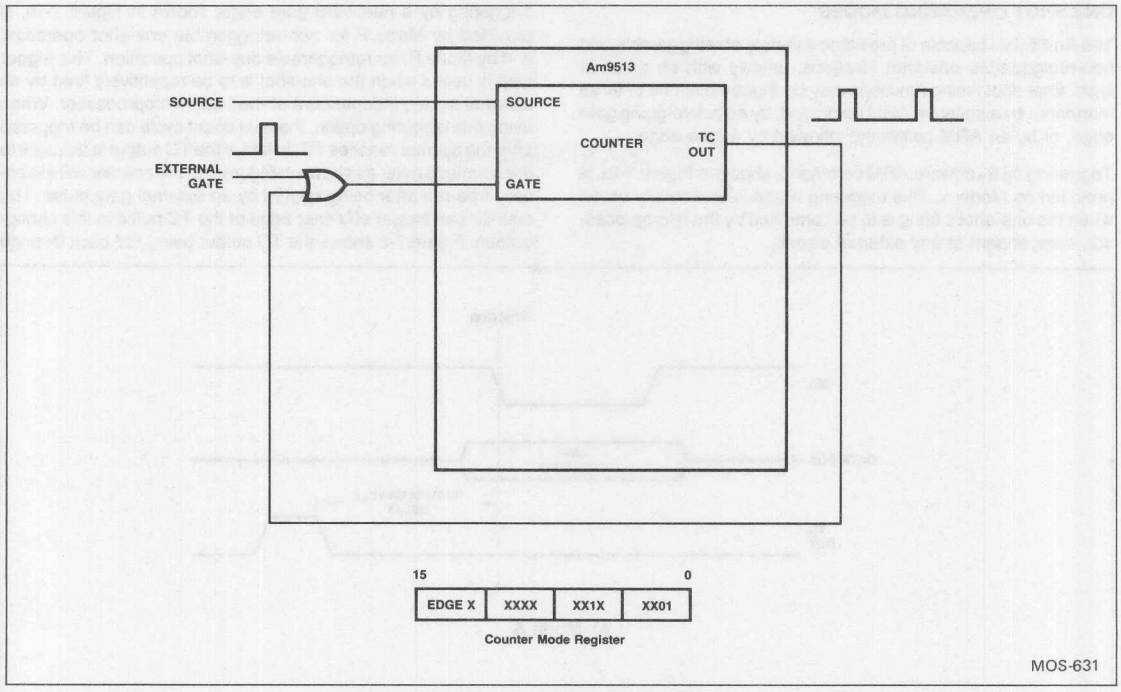


Figure 7-2. TC Feedback Connection

an external OR gate. Applications using an 8-bit data bus can use the AND function provided by the auxiliary Gate inputs as a negative logic OR, eliminating the need for an external gate.

The third trigger type, shown in Figure 7-1c, will fire the one-shot when a software ARM command and a hardware gate edge are applied in that order. This triggering operation is useful in applications which need the microprocessor to qualify when triggering can occur but which actually trigger the one-shot from an external signal.

In addition to the three triggering modes, there are three different types of one-shot output waveforms that may be generated by the Am9513, as shown in Figure 7-3. The type of output waveform obtained is selected by the particular counting mode used and the programming of the Output Control field in the counter's Mode register.

The strobed or TC one-shot output generates an active pulse for a one clock cycle duration when the counter reaches TC. This waveform is useful for triggering other counters or can be used to set, clear or toggle flip-flops to generate more complex signals.

The second one-shot output, called the traditional output, is useful only when gate edge triggering is used. The traditional output goes active following application of the gate edge and remains active until the counter reaches TC. This waveform is the type found on traditional linear one-shots, hence its name.

Although the Am9513 requires an external flip-flop to generate the traditional one-shot output, it is capable of directly generating a more general-purpose signal which has a quasi-traditional output as a special case. This more general signal is the third waveform type and provides a delayed-pulse one-shot. After the counter is triggered, the counter output remains inactive for some programmable delay. The output then goes active for a programmable period. Note that both the delay to the start of the pulse and the pulse width are separately programmable. With the minimum delay to start-of-pulse of 1 clock cycle programmed, the delayed-pulse one-shot signal becomes similar to the traditional one-shot output. The only difference between these modes is that in the traditional one-shot, the output goes active on the gate edge and in the delayed-pulse one-shot, the output goes active on the second source edge after the gate edge.

In some applications it is desirable to restart the timing sequence from the initial value after the counter starts counting but before it reaches TC. This operation, called retriggering in this document, effectively increases the time to TC. The strobed one-shot mode may be operated in a retriggerable mode in the Am9513 without external hardware. With the addition of a few external devices, retriggerable traditional and delayed-pulse one-shot waveforms can be generated.

A matrix can be constructed illustrating the combination of trigger types, output waveforms and retriggering options available with the Am9513. Such a diagram is presented in Figure 7-4. Since the combination of triggering on an ARM command and retriggering by an external gate is not a reasonable operating configuration, it is not included in the chart.

One-Shot Count Sequences

Because the Am9513 counters always reload on the rising TC edge and always count on the trailing TC edge, one-shot count sequences may be different from what users might intuitively expect. Figure 7-5 shows a typical non-retriggerable one-shot count sequence. Initially the counter contents are 5. When a gate edge is applied, the counter begins counting. On the count source

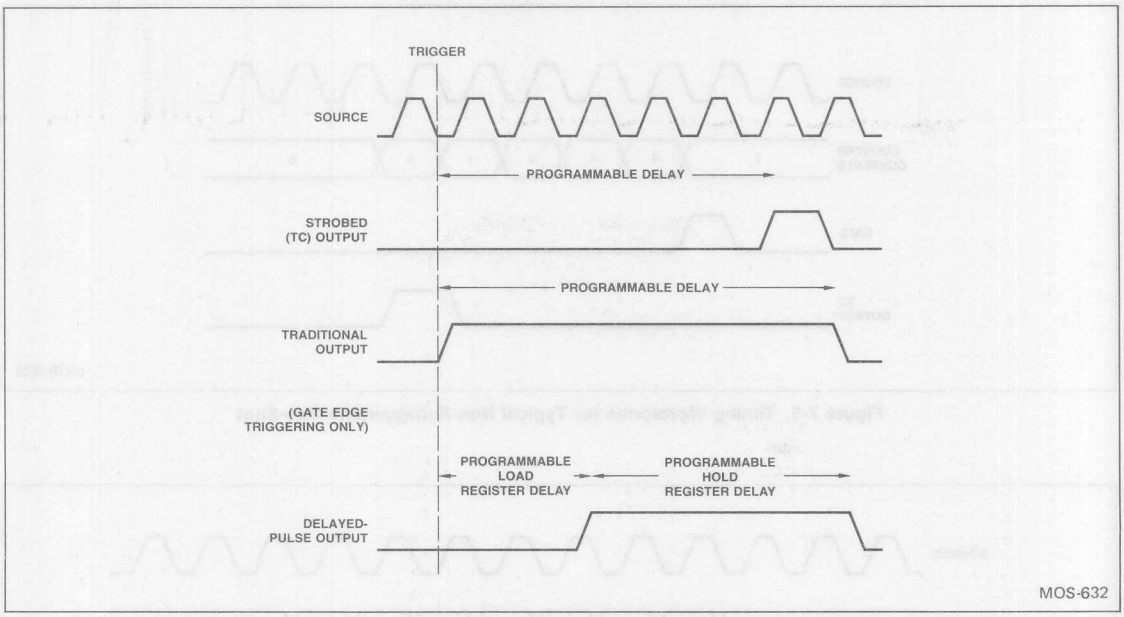


Figure 7-3. One-Shot Output Options

Retrigger Option	Non-Retriggerable			Retriggerable	
Trigger Option	**ARM**	**Gate**	**ARM and Gate**	**Gate**	**ARM and Gate**
Output Options					
Strobed Output	Mode A	Mode F	Mode C	Mode R	Mode O
Traditional Output	Mode A*	Mode F*	Mode C*	Mode R*	Mode O*
Delayed-Pulse Output	Mode G	Mode L	Mode I		

*Additional hardware required.

Figure 7-4. Matrix of Am9513 One-Shot Modes

edge that occurs when the counter contents are 1, the counter goes into TC and reloads itself from the Load register, which contains the value 6. On the trailing TC edge the counter decrements the 6 to 5 and then awaits a subsequent gate edge. Note that because the counter counts on the trailing TC edge, although the Load register contents were 6, in fact only 5 counts occur between the gate edge and the end of TC. Another more subtle point is that on initialization the user should load the Load register, execute the LOAD command and then issue a STEP command before arming the counter. The STEP command will increment/decrement the counter contents by 1 to mimic the reload-and-count-once operation which occurs on TC. This ensures that 5, not 6, will be in the counter when it is triggered by the gate edge and ensures that the time from the first gate edge to the first TC will be the same as the time from subsequent gate edges to subsequent TCs.

Count Sequences may also differ from what a user would expect when a counter is operated in a retrigger mode (Modes N, O, Q and R). In these modes, each active-going gate edge applied to the counter will transfer the counter contents into the Hold register. On the first source edge following the gate edge, the Load register contents will be transferred into the counter. Counting will occur on the second source edge after the gate edge. Note that the first gate edge applied to the counter will both start the counter and trigger a save/reload sequence. Accordingly, there will be a two count difference in retriggerable one-shot count sequences vis-a-vis non-retriggerable one-shot sequences. Figure 7-6 shows a retriggerable one-shot count sequence and can be compared to the non-retriggerable sequence shown in Figure 7-5. Since the retriggerable counter uses the first source edge after the gate edge to reload the counter, it reaches TC two clock cycles later than a non-retriggerable counter with the same Load register contents.

Non-Retriggerable One-Shots

The non-retriggerable strobed one-shot is the most basic one-shot timing mode. In this mode the output goes active for one clock cycle when the counter reaches TC. This mode is generated by selecting a TC output in the counter's output control field. For hardware gate triggering of the one-shot process, counting Mode C or F should be used. In Mode C, a new ARM command followed by a gate edge must be issued to the counter to start

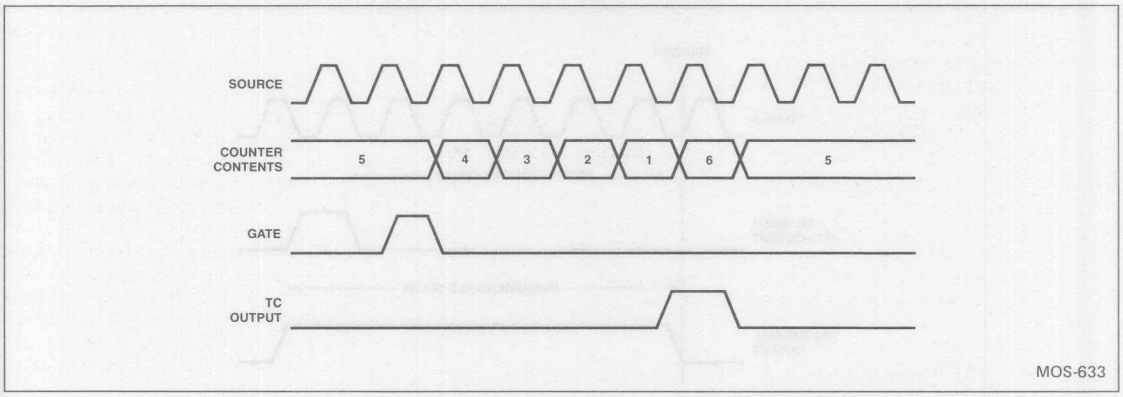


Figure 7-5. Timing Waveforms for Typical Non-Retriggerable One-Shot

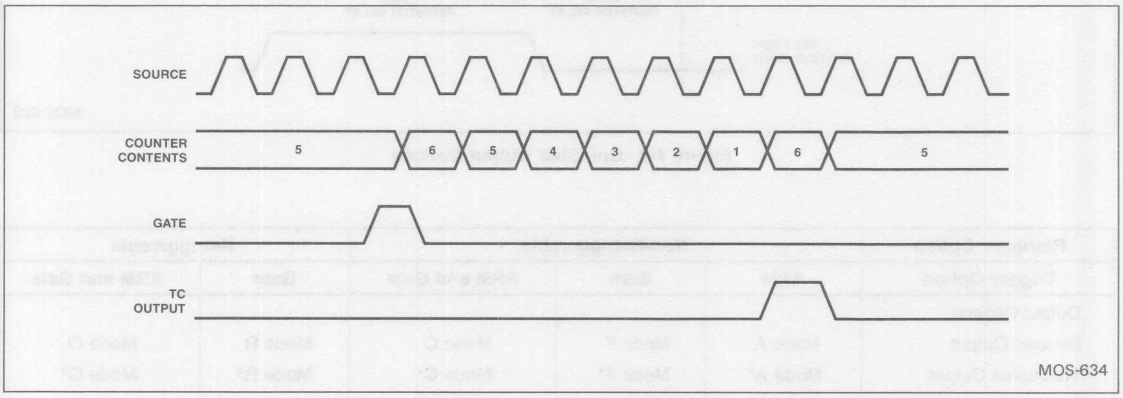


Figure 7-6. Timing Waveforms for Typical Retriggerable One-Shots

each one-shot cycle. In Mode F, the counter will perform one-shot cycles each time a gate edge is applied without requiring a new ARM command for each count cycle. In both Modes C and F, if a gate edge is applied during the count cycle before TC goes active, the edge is disregarded. Mode A provides the third variation of strobed one-shot timing. In this mode, the software ARM command is used to trigger the one-shot operation and the hardware gate input is disregarded. Strobed one-shot functions are useful in marking a particular point in time. The output will often be used to set, clear or toggle flip-flops.

With a minimal amount of external logic, an Am9513 counter can emulate a non-retriggerable, linear one-shot generating the traditional output, as shown in Figure 7-7. The advantages of digital one-shots over their linear counterparts are digital one-shot's high resolution, stable operation and their ability to easily interface with microcomputers. In the circuit shown in Figure 7-7, the counter is programmed for down counting in Mode F. An active-low TC output mode is selected. When a gate edge is applied to the counter, the flip-flop clears and the counter starts counting. When TC is reached, the output goes low for one source period, setting the flip-flop. The time that the flip-flop's Q output is high is controlled by the Load register contents. For a Load register value of K, the output high time is given by (K−2). Note that for this circuit the output goes active after application of the gate edge and is driven inactive by a source edge. The uncertainty in the relationship between the gate and the source gives a maximum count error of 1 count in the output active duration. (This uncertainty arises because the gate edge may occur anytime between shortly after a source edge and shortly before the next source edge.) Use of large count values can reduce the percentage of uncertainty to minimal levels. The delayed-pulse output mode can be programmed to drive the output active on the second source edge after the triggering gate edge. In addition to removing the uncertainty regarding the output active duration, this alternative method has the added advantage of not needing external logic.

Delayed-pulse one-shots can be triggered by software ARM command (Mode G), by hardware gate edge (Mode L) or by any ARM command followed by a gate edge (Mode I). For all these modes, the counter's output control field should be programmed for a "TC Toggled" output (CM2-CM0 = 010). If a TC output is programmed (CM2-CM0 = 001 or 101) a dual-pulse one-shot function may be generated. Here, rather than toggle the output on each TC, a one clock period wide pulse is output. The delay from the trigger to the first TC is controlled by the initial counter contents, which are usually set by a reload from the Load register at the end of a previous timing cycle. The delay between the two TC pulses is controlled by the Hold register contents.

Retriggerable One-Shots

Retriggering of one-shots provides a means to extend the time to TC after the one-shot has started timing and is accomplished by applying a retrigger signal to the counter. The retriggering may be done by hardware means, using the gate input, or by software means, using the LOAD command.

Retriggerable one-shots can be initially triggered by a gate edge (in Mode R) or by an ARM command followed by a gate edge (in Mode O). Note that the triggering gate edge also retriggers the counter; see the "One-Shot Count Sequences" section of this chapter for additional details. In both of the above modes, application of a gate edge once the counter is counting will extend the time to TC by reloading the counter from the Load register. (The counter contents are also saved in the Hold register before the reload operation, but this is not relevant to the retriggering operation.) In either mode the counter can be used in a TC output mode (CM2-CM0 = 001 or 101) to generate a strobed output, or may be used with a flip-flop as shown in Figure 7-7 to generate a traditional output.

One-shot functions may also be generated by software without hardware retriggering. A counter operated in Mode A will perform one count cycle each time it is armed. If the ARM command is viewed as a software trigger, this mode operates like a software triggered one-shot. LOAD commands can be issued to a counter operating in Mode A to extend the time to TC. Here the LOAD command behaves like a software retrigger. In fact, the LOAD command can be used with any of the one-shot modes discussed earlier, excluding the delayed-pulse one-shot mode, to act as a software retrigger, extending the time to TC. The delayed-pulse one-shot mode cannot be retriggered with a LOAD command, because it will reload from the location to be used on the upcoming TC rather than the last TC, which is not a retrigger function.

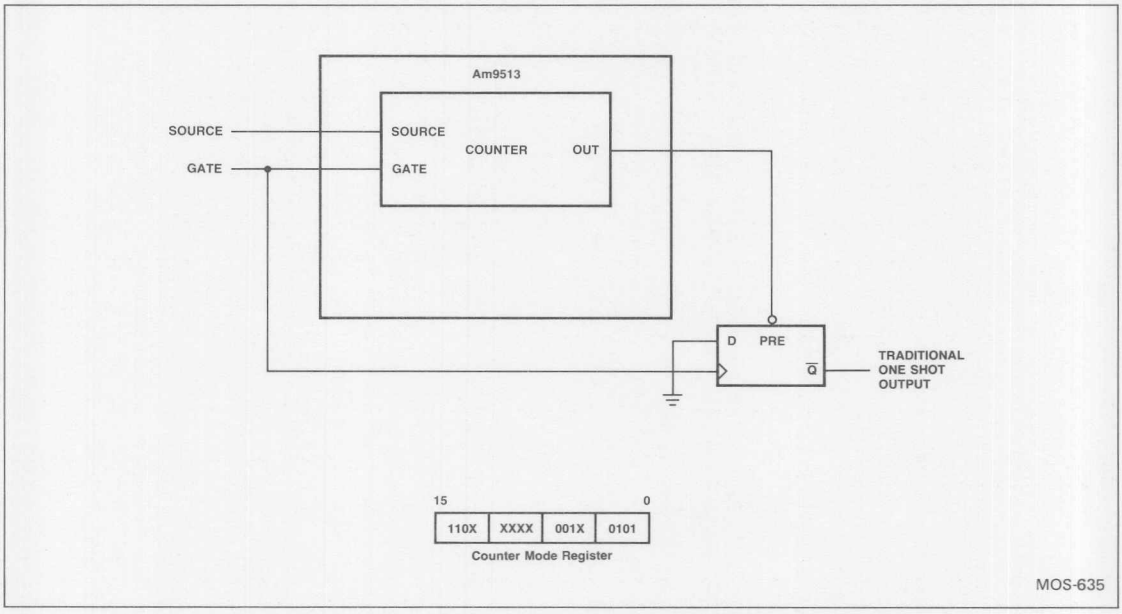


Figure 7-7. Generating a Traditional One-Shot Output

Chapter 8
Software Considerations and Program Examples

Chapter 8
Software Considerations and Program Examples

SOFTWARE CONSIDERATIONS AND PROGRAM EXAMPLES

A device with the versatility of the Am9513 is necessarily complex. This software application section simplifies the programmer's view of the Am9513 and facilitates quicker understanding and the implementation of the facilities provided. Hardware considerations are generally not discussed; indeed, with few exceptions they are deliberately excluded. For a detailed discussion of system implementation with the Am9513, the reader is referred to earlier chapters of this manual.

Example Languages

Software application notes generally provide specific examples of code written in one or more languages, structured as much to illustrate the purpose of the code clearly as well as to provide compact, working programs. Most languages have drawbacks, a fact that is of little importance in an application note, although it is desirable that examples should be presented in both high-level and assembly-level languages.

This software application note presents high-level constructs and examples in the C language, which now is seeing a fairly wide acceptance and is currently supported on systems supplied by Advanced Micro Devices. C is a block-structured language that, although lacking in features such as strong type checking, provides excellent bit-manipulation facilities in record fields, allowing such statements as:

```
#define OFF 1
Master.FOUT_gate = OFF;
```

This statement is part of a record initialization that flags the FOUT gate as being disabled. The "#define" statement assigns the value "1" to the constant identifier "OFF". The bit field ".FOUT_gate" of the record structure "Master" is then assigned the value of the identifier "OFF", i.e., "1". A similar Z8000* assembler listing might take the form:

```
CONST   OFF = 0X1000 ;  % AMD Mnemonics

LD   R4, OFF          ;
OR   R4, MASTER       ;
LD MASTER,R4          ;
```

Throughout this note several C terms have been redefined. Many programmers are familiar with languages other than C, possibly of the Pascal or PL/X variety, so the following definitions have been used in an attempt to achieve some common ground:

```
#define BEGIN
#define END
#define RECORD  struct
#define THEN
```

These definitions allow the use of the type identifier "RECORD" for structure definitions rather than the normal C type identifier "struct".

Assembly language examples are presented using Am8080/Am8085, Z80 and Z8000 source mnemonics having formats compatible with the AMD or Zilog assemblers. Macros are used wherever it is felt that they aid the programmer, mostly within the Am9513 command structure. The macros are generally optimized, such that better code can not be generated by hand, and mostly employ strict parameter checking (eliminating excuses for not using them). For example, to save counters 3, 4 and 5 would be accomplished by the statement:

```
SAVE  3,4,5
```

Code generated by this statement would be (Am8080/Am8085):

```
MVI   A,0BCH
OUT   CONTROL
```

Macros are presented for setting the Counter registers and the Master register and, although the experienced Am9513 user may find them of dubious use, the novice will appreciate the code generated. For example, to set the Master register, the following command may be used:

```
MASTER  TOD_50HZ,DISABLE,DISABLE,GATE_1,
       11,ON,BUS_8,ON,BCD
```

The following code sequence will be generated (Am8080/Am8085):

```
MVI   A,17H
OUT   CONTROL      ;  Point to Master reg
MVI   A,61H
OUT   DATA         ;  Send Low command byte
MVI   A,8BH
OUT   DATA         ;  High command byte
```

By examining the code generated (as above) by the macro assembler, the user may gain quick understanding of the usage of the Am9513. (Notice that the above assembly code does not use intermediate storage records for the data. Use of assembly language often implies high efficiency or compact code is required.) C examples have been compiled to Am8080/Am8085

code using the Whitesmith's C compiler and Z8000 code using the AMD C cross compiler, both running on the AMD System 8/8. All executable Am8080/Am8085 and Z80 target code has been tested using an AMD System 8/8 with an Am95/5032 ROM/EPROM board installed, supporting an Am9513 with control and data ports decoded at I/O addresses DAH and D8H, respectively. All executable Z8000 code has been tested using the above system with an Am96/4116 Z8000 bus master card installed.

FUNCTIONAL DESCRIPTION

(See CHAPTER 1 for detailed description)
The Am9513 includes five general purpose 16-bit counters. A variety of internal frequency sources and external pins may be selected as inputs for individual counters with software selectable active-high or active-low input polarity. Hardware gating of each counter is available. Each counter provides either pulsed or level as well as tri-state and fixed low outputs. The counters can be individually programmed to count up or down in BCD or binary modes. The accumulated count may be read without disturbing the counting process. Any of the counters may be internally concatenated to form an effective counter length of up to 80 bits.

Associated with each counter are a Load register and a Hold register. The Load register automatically reloads the counter to a predefined 16-bit value, thus controlling the effective count period. The Hold register acts either as a second 16-bit Load register for complex waveform generation or as a 16-bit storage register to save count values without disturbing the counting process, thereby permitting the host processor to read intermediate count values.

Two counters have additional Alarm registers and comparators with associated logic to allow operation in a 24-hour time-of-day mode with alarm facility. Clocking may be either in real time or programmed over the full dynamic clocking range of the Am9513.

Each of the five counters has a dedicated output pin that may be programmed to provide a variety of outputs. General-purpose counter inputs are available for configuration under software control, allowing dynamic reassignment of inputs with the facility, for example, to use a single gate pin simultaneously as a clock input to one counter and as a gate input to another.

HARDWARE CONSIDERATIONS

Prefetch

In order to minimize the read access time to internal Am9513 registers, a prefetch circuit is used for all read operations through the Data port. Following each read or write operation through the Data port, the Data Pointer register is updated to point to the next register to be accessed. Immediately following this update, the new register data is transferred to a special prefetch latch at the interface pad logic. When the user performs a subsequent read of the Data port, the data bus drivers are enabled, outputting the prefetched data on the bus. Since the internal data register is accessed prior to the start of the read operation, its access time is transparent to the user. In order to keep the prefetched data consistent with the Data Pointer, prefetches are also performed after each write to the Data port and after execution of the "Load Data Pointer" command. The following rules for Data port Transfers should be heeded:

1. The Data Pointer register should always be reloaded before reading from the Data port if a command other than "Load Data Pointer" (point to or POINT) was issued to the Am9513 following the last Data port read or write. The Data Pointer does not have to be loaded again if the first Data port transaction after a command entry is a write, since the Data port write will automatically cause a new prefetch to occur.

2. Operating modes N, O, Q, R and X allow the user to save the counter contents in the Hold register by applying an active-going gate edge. If the Data Pointer register had been pointing to the Hold register in question, the prefetched value will not correspond to the new value saved in the Hold register. To avoid reading an incorrect value, a new "Load Data Pointer" command should be issued before attempting to read the saved data. A Data port write (to another register) will also initiate a prefetch; subsequent reads will access the recently saved Hold register data. Many systems use the "saving" gate edge to interrupt the host CPU. In such systems, the interrupt service routine should issue a "Load Data Pointer" command prior to reading the saved data.

Memory Mapping

Associated with the Am9513 are four parameters known collectively as read/write recovery times. Certain coding sequences can violate these parameters in sometimes non-obvious ways. Consider a Motorola 6800-based system, employing memory-mapped input/output. Reading the hold register on counter 4 may be accomplished by the following code sequence:

```
LDAA    # 14       Point to counter 4 hold
STAA    CONTROL
LDX     DATA       Read the register
STX     RESULT     Store the data
```

Notice that the operation "LDX" performs two 8-bit reads from location DATA and DATA+1 using two consecutive clock cycles. For a 125Ons read recovery time the maximum system clock frequency allowable without violating the read recovery time is 680kHz. This is lower than many system clocks in use, so the following code sequence may be used to avoid this limitation:

```
LDAA # 14       Point to counter 4 hold
STAA CONTROL
LDAA DATA       read low byte
STAA RESULT     reversed order for
                compatability with above

LDAA DATA+1     read high byte
STAA RESULT
```

The equivalent multiple write operation is even more restrictive: an 1800ns write recovery time limiting the system clock to 500kHz unless the separate data byte write technique is adopted.

Similar considerations apply for memory-mapped Am8080/Am8085 systems. Notice that in the case of normal I/O mapped Z80 and Z8000 systems, the block input/output instructions may also violate the read/write recovery time parameters at very high system clock rates. Using the normal input/output instructions avoids violations without recourse to the use of wait states.

DATA MODEL

The first task of the programmer is to construct a data model of the Am9513 around which the applications software may be draped. A top-down, structured approach is adopted and although for purposes here the data model need not be optimized either for space utilization or for access time, both aspects are in fact efficiently implemented.

Figure 8-1 shows the data model to be adopted, which should be contrasted with Figures 8-2, 8-3 and 8-4 depicting the hardware structure being modeled. Notice that the Command register and Data Pointer register do not appear in the data model.

The simplicity of Figure 8-1 illustrates that the Am9513 is singularly well suited to this approach; removing extraneous internal hardware aspects focuses the attention of the programmer on the salient details.

C provides a facility called "typedef" for creating new data type names, similar to the "TYPE" facility in other languages. For example, the declaration

```
typedef int LENGTH;
```

makes the name 'LENGTH' a synonym for 'int'. The type 'LENGTH' can be used wherever the type 'int' can be used. Similarly, the two record definitions of Figure 8-1 can be replaced by their respective type names; indeed, the definition of the 'Am9513_type' uses the type 'channel_type' as a field to identify the array of five counters on the Am9513. Notice from Figure 8-1 that each counter (or channel_type) consists of three registers known as the Mode, Load and Hold registers. Additionally the Mode register itself is further defined via a typedef in Figure 8-**6** to consist of a series of bit fields. The sum of the field widths of these bit fields is 16, that is, the Mode register is 16 bits wide.

It must be emphasized that a C typedef declaration does not create a new type in any sense (unlike the TYPE declaration in other languages); it merely adds a new name for some existing type. The use of typedef's makes C programs more portable and significantly improves readability.

```
                                    /* Single counter set    */

typedef RECORD  {count_type         mode ;
                 unsigned int       load ;
                 unsigned int       hold ;
                } channel_type ;

                                    /* Am9513 chip set       */

typedef RECORD  {master_type        master ;
                 channel_type       counter [5] ;
                 status_type        status ;
                 unsigned int       alarm_1 ;
                 unsigned int       alarm_2 ;
                } AM9513_type ;
```

Figure 8-1. Software Structure of the Am9513

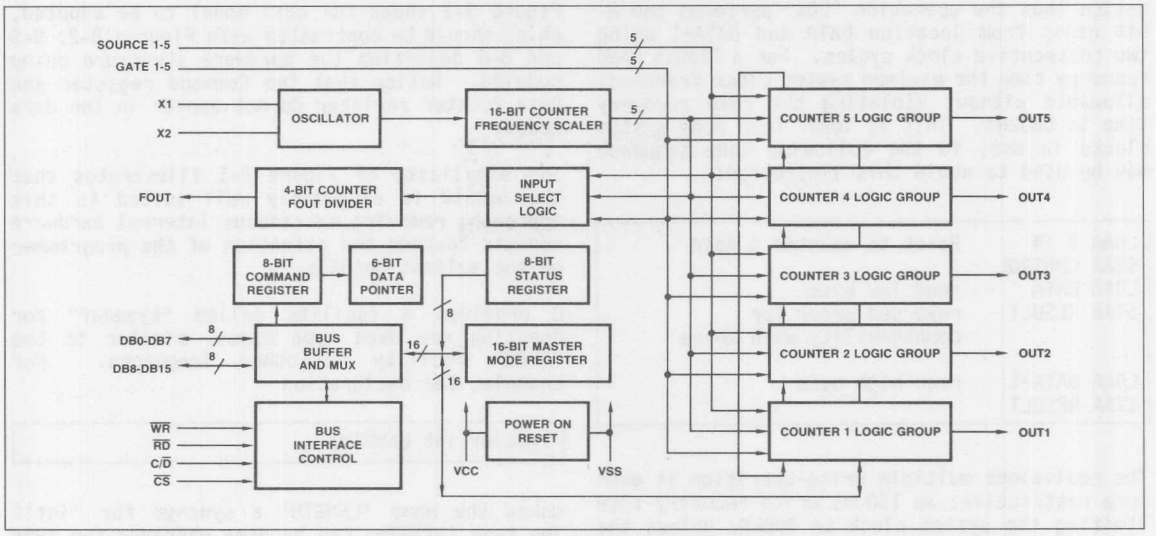


Figure 8-2. General Block Diagram

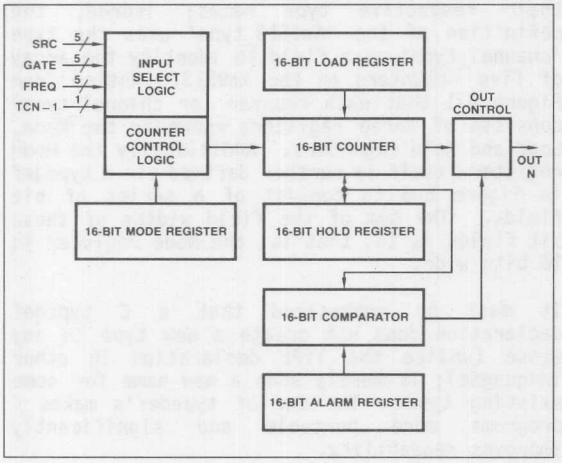


Figure 8-3. Counter Logic Groups 1 and 2

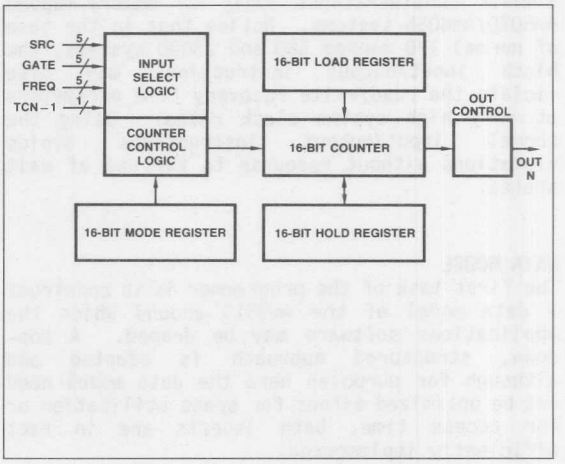


Figure 8-4. Counter Logic Groups 3, 4 and 5

An application using an array of 20 Am9513 devices with an access pointer may be declared as shown in Figure 8-5a.

Notice that C uses array subscripts commencing with zero; the third Am9513 device would be referred to by Am9513[2]. Should the reader prefer, all arrays can be declared with a dummy entry, such that all zero indices may be ignored as shown in Figure 8-5b.

```
Am9513_type
                /* recall we have a new "type"*/
Am9513[20],     /* data structure for 20 chips*/
*Am9513_ptr  ;  /* access pointer */
```

Figure 8-5a.

The third Am9513 device may now be referenced by the form Am9513[3]. Throughout this software application note the first form (zero subscript allowed) will be employed except where clearly stated otherwise.

The following discussions of the data model fields (the various hardware registers on the Am9513) are accompanied by examples of usage. The examples refer to the model, which is

```
Am9513_type
Am9513[21], /* 21 devices, ignore device #0 */
```

Figure 8-5b.

transferred to the hardware device via some input/output operations. Such operations are discussed with the data pointer sequencing facility description, and are omitted from most other C examples for clarity.

Users of other languages should find the data model and C examples useful as an aid to understanding the Am9513. Assembly language examples are included for additional clarity.

COMMAND REGISTER

The Command register provides direct control over each of the five general counters (via the Control port) and controls access to the counter registers (via the Data port) by updating the Data Pointer register. Commands are instruction codes to the Am9513 and as such are not part of the data model. They are generally used in the form:

```
eg ()
BEGIN
     int reset = OXFF,          /* C code */
         load_all_counters = OX5F;

     output  (CONTROL, &reset);
     output  (CONTROL, &load_all_counters);
END
```

Notice that the variables "reset" and "load_all_counters" are declared AUTO (LOCAL in some languages) such that they reside on the stack while the procedure "e.g.()" is current. The operation of the procedure "output (X,Y)" will be described later in this section; suffice to say that the address of the variable "Y" to be output (sent to the Am9513) is passed as one parameter and the destination port "X" (Control or Data) of the Am9513 as the other. Thus the variable "Y" is called by reference (e.g., &reset) and the destination port "X" is called by value (e.g., Control).

```
                    ; Z80 Macro Assembler code
ENTRY:
     RESET          ; Macro simplicity
     RET
```

The "Command Description" section later in this chapter explains the detailed operation of commands available with example usage.

DATA POINTER REGISTER

The Data Pointer register is a write-only register controlled solely by a command with the structure illustrated in Figure 8-6. The detailed hardware format of the register is irrelevant for purposes here since the data pointer command provides all information necessary. As a command, it does not appear in the data model.

```
     /* Load data ptr register command */
typedef RECORD {unsigned group       :3;
                unsigned element     :2;
                unsigned cmnd_code   :3;
                }data_type ;
```

Figure 8-6. Load Data Pointer Command Structure

The data pointer command selects which internal register is to be accessible via the Data port and consists of a constant command code (000), a 2-bit element field and a 3-bit group field. (See Page 1-5 for hardware description.) Random access to any available internal register location can be accomplished by simply sending the appropriate data pointer command to the Control port and then performing a Data read or write. Sequential access to groups of internal registers may be performed by sending the appropriate enable and data pointer commands to the Control port and performing multiple Data reads and/or writes.

For example, random access to the Load register of counter 4 may be performed as shown in Figure 8-7. Sequential access to the hold registers of all five counters is performed as shown in Figure 8-8. Suitable C input/output routines for an 8-bit data bus may be defined as shown in Figure 8-9.

MASTER MODE REGISTER

The 16-bit Master Mode Register controls those internal activities that are not controlled by the individual counter registers. Figure 8-10 shows the record fields of the 'Master-type' structure. Figure 8-11 illustrates allowable field values.

The individual field declarations show, for example, that the '.FOUT_source' field is four bits wide. Notice that the sum of all the field widths is 16; the Master Mode Register is 16 bits wide.

C compilers should be checked to ensure bit field declarations are implemented with first field at the least significant bit address. These definitions are correct for the AMD and Whitesmith's compilers; some other compilers may need the field order reversed.

```
                                                    /* C example */
eg ()
BEGIN
      point_to (4,LOAD)  ;                          /* set up Data Pointer Register */
      Am9513.counter[3].load = input(DATA)  ;       /* Read in the data to appropriate
                                                                        field */
END


point_to (channel, reg)                             /* C utility  */
      unsigned int channel, reg  ;                  /* set Data Pointer Register to
                                                                    channel,reg */
BEGIN
      data_type data_ptr  ;                         /* local command structure */

      data_ptr.group = channel  ;
      data_ptr.element = reg  ;
      data_ptr.cmnd_code = 0  ;
      output (CONTROL, &data_ptr)  ;                /* send Load Data Pointer Command */
END


                                                    ; Z80 Macro example
STORE DEFS   2                                      ; Destination of data

ENTRY:
      POINT       4,LOAD_                           ; Point to counter 4 LOAD register
      LD          C,DATA                            ; Set up port address
      LD          HL,STORE                          ; Set up data destination
      INI                                           ; Low byte of LOAD register
      INI                                           ; High byte of LOAD register
      RET
```

Figure 8-7. Random Access to Registers

```
                                                    /* C example */
eg ()
BEGIN
      int index = -1  ;                             /* counter index, 0 to 4   */

      sequence (ENABLE)  ;                          /* Turn on data pointer seq   */
      point_to (1,HOLD_CYCLE);                      /* Set Data Pointer register  */
      while ((index +=1) <5)                        /* Counters 0 thru 4   */
            Am9513.counter[index].hold = input(DATA);  /* 16 bit transfer */
END

sequence (request)                                  /* C utility   */
      int request ;
BEGIN
      int enable = 0XE0 ,                           /* commands for data pointer sequencing  */
          disable = 0XE8    ;

      if (request == ENABLE)
            output (CONTROL, &enable)  ;
      else
            output (CONTROL, &disable)  ;
END
```

Figure 8-8. Sequential Access to Registers

```
                                        ; Z80 Macro example
ENTRY:
      DPS  ON                           ; Enable data pointer sequencing
      POINT 1,HOLD_CY                   ; Data pointer to hold cycle
      LD  HL,HOLDS                      ; Data area
      LD  B,5*2                         ; Byte count, 5 holds, 2 bytes each
      LD  C,DATA                        ; Am 9513 data port
      INIR                              ; Perform transfer
      RET

HOLDS: DEFS 10                          ; Data area

                                        ; Z80 Macro example
ENTRY:
      DPS  ON                           ; Enable data pointer sequencing
      POINT 1,HOLD_CY                   ; Data pointer to hold cycle
      LD  HL,HOLDS                      ; Data area
      LD  B,5*2                         ; Byte count, 5 holds, 2 bytes each
      LD  C,DATA                        ; Am 9513 data port
      INIR                              ; Perform transfer
      RET

HOLDS: DEFS 10                          ; Data area

                                        ; Am8080/Am8085 Macro example
      DPS    ON                         ; Enable data pointer sequencing
      POINT  1,MODE_                    ; Mode register 1
      LXI    H,ARRAY                    ; Storage area
      MVI    B,5*3*2                    ; 5 counters, 3 x 16 bit registers
LOOP:
      IN     DATA                       ; Get a byte
      MOV    M,A                        ; Save a byte
      INX    H                          ; Move the byte pointer
      DEC    B
      JNZ    LOOP
      RET

ARRAY:
MODE1 DS     2                          ; counter 1 data area
LOAD1 DS     2
HOLD1 DS     2
...
MODE5 DS     2                          ; counter 5 data area
LOAD5 DS     2
HOLD5 DS     2
```

Figure 8-8. Sequential Access to Registers (Cont.)

After power-on reset or a Software Reset command the Master Mode Register is set to the following configuration (all field values zero):

```
.day-mode      = TOD_OFF
.compar_1      = DISABLE
.compar_2      = DISABLE
.FOUT_source   = F1
.FOUT_divisor  = 16
.FOUT_gate     = ON
.data-bus      = BUS_8
.data-ptr      = ON
.scaler        = BINARY
```

Notice that changing the FOUT status by altering the source divisor or gate fields may generate transients.

Time-Of-Day

The .day_mode field can be turned off, allowing counters 1 and 2 to function the same way as counters 3, 4 and 5, or set to TOD_50HZ,TOD_60HZ or TOD_100HZ allowing a 24-hour clock function to be used. Refer to the software examples (in Chapter 4) for further information.

```
                                        /* C example */

unsigned int input (port)               /* Read 2 bytes from port */
       int port  ;
BEGIN
      unsigned int temp;               /* local parameter to assemble word */

      temp = in (port) ;                /* Get low byte */
      temp += in (port) *256  ;         /* Put high byte into temp */
      return (temp)  ;                  /* Return the 16 bit value read */
END

output (port, data)                     /* 2 bytes to DATA port or 1 byte */
      unsigned port, *data;             /* to CONTROL port (8 bit bus) */
BEGIN
      out (port, (*data %256));
      if  (port == DATA)                /* 16 bit transfer to DATA port */
          out (port, (*data/256));
END
```

Figure 8-9. Input/Output Routines for 8-Bit Data Bus

```
typedef RECORD      {unsigned day_mode     :2;
                     unsigned compar_1     :1;
                     unsigned compar_2     :1;
                     unsigned FOUT_source  :4;
                     unsigned FOUT_divisor :4;
                     unsigned FOUT_gate    :1;
                     unsigned data_bus     :1;
                     unsigned data_ptr     :1;
                     unsigned scaler       :1;
                    }master_type;
```

Figure 8-10. Master Mode Register – Software Structure

Comparators

Comparator registers exist for counters 1 and 2. If a comparator is enabled (e.g., .compar_1 = ENABLE), its output is substituted for the associated counter output. The output will remain active while the comparison is true. The two comparators can always be used individually in any operating mode. One special case occurs when the Time-of-Day option is invoked and both comparators are enabled. The operation of Comparator 2 will then be conditioned by Comparator 1 so that a full 32-bit compare must be true in order to generate a true signal on Output 2. Output 1 will continue, as usual, to reflect the state of the 16-bit comparison between Alarm 1 and Counter 1.

FOUT Source

The '.FOUT_source' field specifies the source input for the FOUT divider. Fifteen inputs are available for selection including the five source pins (SRC1-SRC5), the five Gate pins (GATE1-GATE5) and the five internal frequencies derived from the master oscillator (F1-F5).

```
e.g.,  Am9513.master.FOUT_source = GATE_4;
```

FOUT Divider

The '.FOUT_divisor' field specifies the dividing ratio for the FOUT Divider. The .FOUT_source is divided by an integer value (1-16) and passed to the FOUT output buffer.

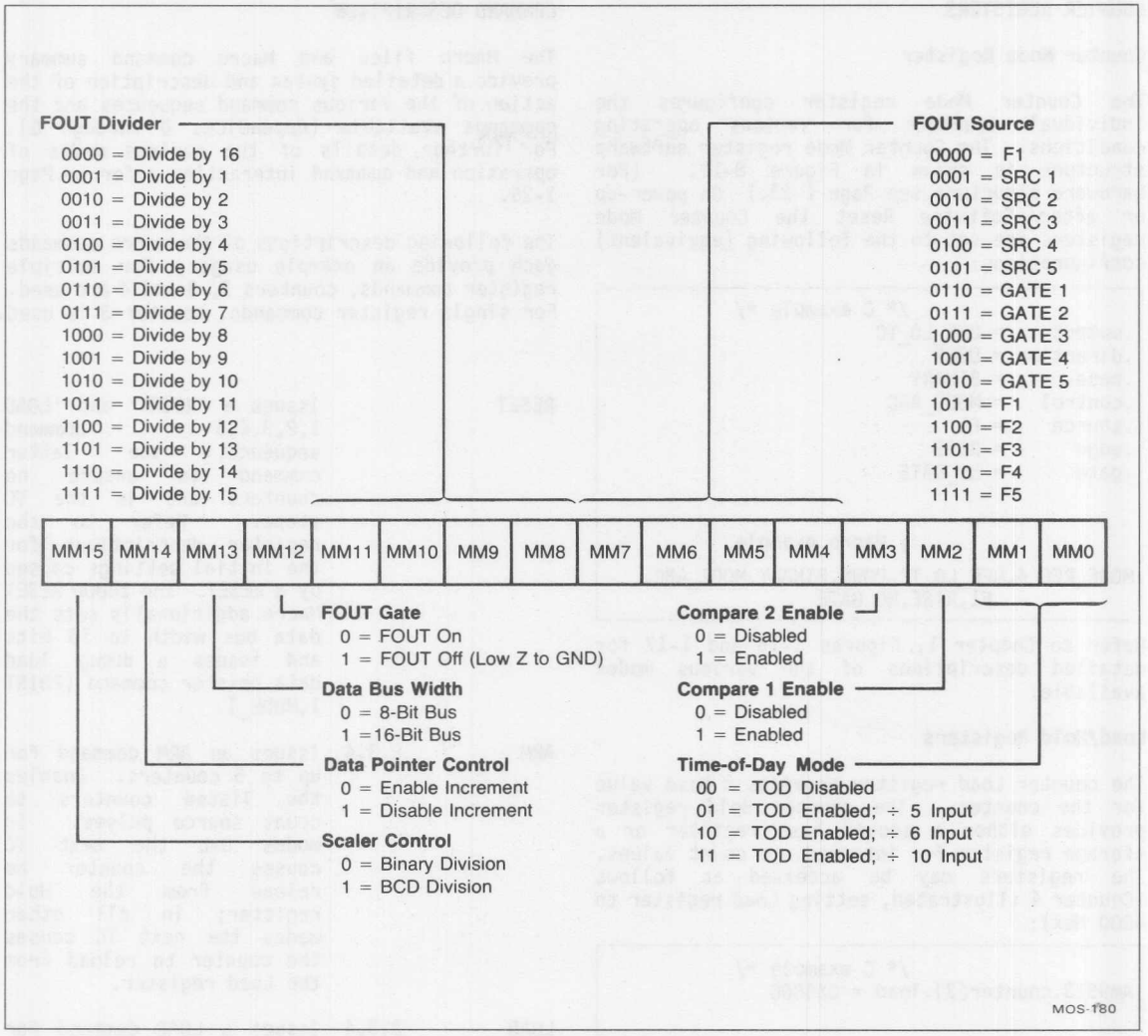


Figure 8-11. Master Mode Register – Hardware Structures

FOUT Gate

The '.FOUT_gate' provides a software gating facility for the FOUT output signal (ON or OFF). Notice that commands exist to directly gate the FOUT output on and off without using the Master register fields (e.g., FOUT ON).

Bus Width

The 'data_bus' field controls the width of the data bus interface by configuring the part for an 8-bit or 16-bit external data bus. Notice that the CP/M-compatible version of the Am9513 evaluation program and the Am8080/Am8085 and Z80 macros only allow an 8-bit data bus configuration while the Z8000 macros only allow a 16-bit configuration. These constraints may be changed by the user. Notice that the macros require the bus width to be passed as a parameter, although it is ignored.

Data Pointer Sequencing

The '.data_ptr' field enables or disables the automatic sequencing functions, described under the Data Pointer Register section. Commands exist to directly enable or disable this function (e.g., DPS ON).

Scaler Ratio

The '.scaler' field controls the counting configuration of the frequency scaler counter. This configuration may be BCD or BINARY.

COUNTER REGISTERS

Counter Mode Register

The Counter Mode register configures the individual counters for various operating conditions. The Counter Mode register software structure is shown in Figure 8-12. (For hardware structure see Page 1-23.) On power-up or after Software Reset the Counter Mode registers are set to the following (equivalent) configurations:

```
                    /* C example */
.output    = OFF_LO_TC
.direction = DOWN
.base      = BINARY
.control   = MODE_ABC
.source    = F1
.edge      = RISE
.gate      = NO_GATE


                    ; Macro example
MODE_REG 4,OFF_LO_TC,DOWN,BINARY,MODE_ABC,
         F1,RISE,NO_GATE
```

Refer to Chapter 1, Figures 1-16 and 1-17 for detailed descriptions of the various modes available.

Load/Hold Registers

The counter Load register provides a base value for the counter. The counter Hold register provides either a second base register or a storage register for intermediate count values. The registers may be accessed as follows (Counter 4 illustrated, setting Load register to 4000 Hex):

```
                    /* C example */
Am9513.counter[3].load = 0X4000


                    ; Macro example
LOAD_REG   4,4000H
```

COMMAND DESCRIPTION

The Macro files and Macro command summary provide a detailed syntax and description of the action of the various command sequences and the commands available (Appendices D through G). For further details of the various modes of operation and command interaction refer to Page 1-25.

The following descriptions of the macro commands each provide an example usage. For multiple register commands, counters 2, 3 and 4 are used. For single register commands, counter 3 is used.

RESET		Issues a 'RESET' and 'LOAD 1,2,3,4,5' command sequence, the latter command to ensure no counters are in the TC state. Refer to the register descriptions for the initial settings caused by a RESET. The Z8000 RESET Macro additionally sets the data bus width to 16 bits and issues a dummy load data pointer command (POINT 1,MODE_).
ARM	2,3,4	Issues an ARM command for up to 5 counters. Enables the listed counters to count source pulses. In modes G-L the next TC causes the counter to reload from the Hold register; in all other modes the next TC causes the counter to reload from the Load register.
LOAD	2,3,4	Issues a LOAD command for up to 5 counters. Causes the listed counters to be loaded with the Load or

```
                                                    /* Counter Mode Register */
typedef RECORD      {unsigned output                :3;
                     unsigned direction             :1;
                     unsigned base                  :1;
                     unsigned control               :3;
                     unsigned source                :4;
                     unsigned edge                  :1;
                     unsigned gate                  :3;
                    }count_type;
```

Figure 8-12. Counter Mode Register – Software Structure

		Hold register depending on the Mode register and the state of the count cycle. If a listed counter is in the TC state, the counter counts once to leave TC, before loading, regardless of whether the counter is armed or the state of any gate input. If a listed counter is in the cycle preceding TC, the counter immediately goes to TC, regardless of whether the counter is armed or of the state of any gate input.
LD_ARM	2,3,4	Issues a LOAD AND ARM command for up to 5 counters. Operation is identical to issuing separate LOAD and ARM commands. If a listed counter is in the TC state the counter will count once to leave TC before loading and arming, regardless of whether the counter is armed or of the state of any gate input. If a listed counter is in the cycle immediately preceding TC the counter will immediately go to TC regardless of whether the counter is armed or of the state of any gate input. Avoid this command, by using separate LOAD and ARM commands where possible.
DISARM	2,3,4	Issues a DISARM command for up to 5 counters. If a listed counter is in the TC state, the counter counts one source edge, to leave TC, before disarming. Once disarmed all counting ceases.
SAVE	2,3,4	Issues a SAVE command for up to 5 counters. Transfers contents of listed counters into associated HOLD register independent of, and without affecting counter status.
DRMSAV	2,3,4	Issues a DISARM AND SAVE command for up to 5 counters. Identical to issuing a DISARM and a SAVE command.
SET_	3	Issues a SET command to a single counter. The output for the counter is driven high provided TC toggle mode is specified, otherwise nothing happens.
CLEAR	3	Issues a CLEAR command to a single counter. The output for the counter is driven low provided TC toggle mode is specified, otherwise nothing happens.
STEP	3	Issues a STEP command to a single counter. Increments or decrements the counter irrespective of armed status or gate conditions. The step direction depends on the Mode register. The results of stepping an armed counter while counting are undefined.
FOUT	ON	Issues a GATE ON FOUT or GATE OFF FOUT command. The FOUT output becomes active/inactive. Notice that a single transient pulse may be produced.
DPS	ON	Issues an ENABLE/DISABLE DATA POINTER SEQUENCE command. See the description of the Data Pointer register for further details.
POINT	3,MODE_	Issues a LOAD DATA POINTER REGISTER command with group, element information. See the description of the Data Pointer register for further details.
MASTER	TOD_OFF,DISABLE,DISABLE,F4,15,ON,BUS_8,ON,BINARY	Issues a command sequence to set the Master register. The nine parameters correspond to the nine data model fields and as such are reversed in order, reading low order bits to high order bits from left to right. Notice that the nine parameters are NOT checked by the macros for space reasons.
MODE_REG	3,TC_TOGGLE,UP,BCD,MODE_ABC,F4,FALL,NO_GATE	Issues a command sequence to set the Mode register for a counter. The first parameter identifies the counter and the next seven

parameters correspond to the seven data model fields. The seven data model fields read low order bit to higher order bit from left to right. The seven parameters are NOT checked by the macros for space reasons.

LOAD_REG 3, 4000H

Issues the command sequence to set the Load register for a single counter to a given value, in this case a constant, 4000H.

HOLD_REG 3,4000H,I

Issues the command sequence to set the Hold register of a single counter to a given value, in this case the contents of the address 4000H. The Z8000 macro does not require the indirection flag (I) since the AMD Assembler Macz is able to identify the indirection mode from context.

EXAMPLE: Am9513 EVALUATION PROGRAM

The Am9513 evaluation program is menu-driven allowing full functional testing and evaluation of an Am9513 located within a host system I/O address space. The evaluation program is written entirely in C and is provided under the CP/M (ver 2.2) compatible AMDOS operating system to allow interactive usage without the necessity of a resident C compiler.

The Am9513 driver uses the data model and commands described earlier, thus providing a useful example of Am9513 C programming. Additionally, at certain points throughout the driver program, further explanation is provided on request, mostly in the area of the various operating modes available. All functions associated with the Am9513 may be exercised, including defining the port addresses associated with the device under test. However, since the target code is for the Am8080/Am8085, the data bus width is restricted to 8 bits.

Figure 8-13 provides an example of the coding used within the evaluation program and Figure 8-14 shows an actual user run-time test session (user input is highlighted).

```
Am9513_type
        Am9513,                 /* declare the storage  */
        *Am9513_ptr ;

read_element (mode)

/*      This procedure will read the chip registers using either
SEQUENCE or random-access depending on the mode parameter request.
*/

        int mode ;
BEGIN
        int loop_count ;
        unsigned int A[3] ;             /* temp array for register values */
        int group, element ;            /* indices          */

        if ( mode == SEQUENCE )
        THEN
        BEGIN
                sequence (ENABLE) ;
                point_to (1,MODE) ;
                Am9513_ptr = &Am9513[Ø] ;
                loop_count = Ø ;
                while (( loop_count += 1 ) <6)
                BEGIN
                        Am9513_ptr->counter[group].mode = input (DATA) ;
                        Am9513_ptr->counter[group].load = input (DATA) ;
                        Am9513_ptr->counter[group].hold = input (DATA) ;
                        Am9513_ptr++ ;
                END
        END
        else
        BEGIN
                Am9513_ptr = &Am9513[Ø] ;
                group = Ø ;
                sequence (DISABLE) ;
                while (( group += 1 ) <6 )
                BEGIN
                        element = -1 ;
                        while (( element += 1 ) <4 )
                        BEGIN
                                point_to (group, element) ;
                                A[element] = input (DATA) ;
                        END
                        Am9513_ptr->counter[group].mode = A[Ø] ;
                        Am9513_ptr->counter[group].load = A[1] ;
                        Am9513_ptr->counter[group].hold = A[2] ;
                        Am9513_ptr++ ;
                END
        END
END
```

Figure 8-13. Am9513 Evaluation Program Segment

```
Main Menu
---------
1. Set data bus width              8. Load and arm
2. Software reset                  9. Disarm
3. Set output channel              A. Save
4. Set Master register             B. Disarm and save
5. Set port addresses              C. Step
6. Arm counters                    D. Read registers, etc
7. Load                            E. Set counter operating modes, etc
                                   F. Clear output channel
  <enter option>
D

0. All regs using data ptr sequencing 3. Any regs, random access
1. All regs using random access       4. Any regs, random access, n times
2. More info                          5. Print entire 9513 reg. set
                                      6. Return to main menu
  <enter option>
0

---------- Last written ----------    ---------- Just read ----------

#         mode          load   hold          mode          load   hold TC

1 00001011 00000000 A     0H     0H     00000000 00000000     0H     0H  0
2 00001011 00000000 A     0H     0H     00000000 00000000     0H     0H  0
3 00011100 11101001 V   100H   200H     00000000 00000000     0H     0H  0
4 00001011 00000000 A     0H     0H     00000000 00000000     0H     0H  0
5 00001111 00100010 D   400H     0H     00000000 00000000     0H     0H  0

0. All regs using data ptr sequencing 3. Any regs, random access
1. All regs using random access       4. Any regs, random access, n times
2. More info                          5. Print entire 9513 reg. set
                                      6. Return to main menu
  <enter option>
6

Main Menu
---------
1. Set data bus width              8. Load and arm
2. Software reset                  9. Disarm
3. Set output channel              A. Save
4. Set Master register             B. Disarm and save
5. Set port addresses              C. Step
6. Arm counters                    D. Read registers, etc
7. Load                            E. Set counter operating modes, etc
                                   F. Clear output channel
  <enter option>
6

Arm counters: enter 1-5 on one line
35
Code used 34H
```

Figure 8-14. Am9513 Evaluation Program Session

Appendices

APPENDIX A – DEALING WITH METASTABLE PROBLEMS

Any circuit employing memory storage devices (i.e., flip-flops) is susceptible to metastable problems. These problems will surface when the storage device is clocked in close temporal proximity to changes on the data inputs. For example, with a D-type flip-flop, changing the D input near an active-going clock edge may cause the flip-flop to go into a metastable state. When a device goes into a metastable state, the Q and $\overline{Q}$ outputs may both be at the same level, may be at intermediate levels or may oscillate between levels. The circuit can remain in the metastable state for a time duration many times longer than the normal propagation delay of the device. The time that the metastable state will persist is based on the exponentially decreasing probability curve (see Figure A-1), with the rate of decrease dependent on the gain-bandwidth product of the device.

Circuit devices such as flip-flops have setup and hold time requirements on the control inputs specified relative to applied clock edges. These setup and hold parameters specify the window during which changes on the control lines may cause a metastable. As long as these parameters are met, metastables will not arise.

Computer system designers eliminate most metastable problems by designing synchronous systems which guarantee the required setup and hold time between clock edges and storage device control inputs. Inputs which are inherently asynchronous, such as interrupts, are sampled and held for a sufficiently long time to allow most metastables to die out. Since the probability of a metastable persisting decreases exponentially, it will never reach zero. Designers of systems with asynchronous inputs use the maximum acceptable failure rate as a guide to determining the time allowed for metastables to decay after sampling the asynchronous input signal.

The data sheet parameters pertaining to synchronization for the Am9513 are TGVEH, TEHGV, TEHWH, TWHEH, TGVWH and TWHGV and for the Am8253 are t_{GS} and t_{GH}. As long as these parameters are met, there will be no metastable problems. To determine if a circuit meets these parameters, use the following guidelines.

Parameters TGVEH and TEHGV for the Am9513 and t_{GS} and t_{GH} for the 8253 specify that when a source edge is applied to an armed counter, the gate must be stable TGVEH (Am9513) or t_{GS} (8253) before the source edge and the gate must be held stable for TEHGV (Am9513) or t_{GH} (8253) after the source edge for level gating; and for edge gating meaningful transitions on the gate input must not occur closer than TGVEH (Am9513) or t_{GS} (8253) before the source edge or sooner than TEHGV (Am9513) or t_{GH} (8253) after the source edge. A meaningful transition in edge gating is defined as a gate edge applied to an armed counter that starts the counter counting in Modes C, F, I, L, O or R, (Am9513) or any active-going gate edge applied to an armed counter in Modes O or R. In the 8253, parameters t_{GS} and t_{GH} must be met relative to the falling source edge; in the Am9513, parameters TGVEH and TEHGV should be met relative to the rising source edge if Counter Mode register bit CM12 = 0 and to the falling source edge if CM12 = 1. These two parameters need only be met while the counter is armed.

Parameters TEHWH and TWHEH (Am9513) specify that when a write command is issued to a given counter, that counter must not have an active-going count source edge any closer than TEHWH before the rising $\overline{WR}$ edge and the next active-going count source edge must not occur until TWHEH after the rising $\overline{WR}$ edge. These parameters must be met for all commands issued to an armed counter and for both ARM and LOAD-and-ARM commands issued to a disarmed counter. As an aside, if these parameters can not be met, it is strongly suggested that the programmer use separate LOAD and ARM commands rather than the combined LOAD-and-ARM command in order to limit the number of events that need to be simultaneously performed. The 8253 has a similar restriction, namely that applying a rising $\overline{WR}$ edge to a counter close to our active-going source edge may cause counter problems. However, the 8253's data sheet, unlike the Am9513's, does not call out what constitutes a safe application of a write to counter.

Parameters TGVWH and TWHGV (Am9513) specify that when a write command is issued to a given counter, the gate input to that counter must be stable TGVWH before the rising $\overline{WR}$ edge and the gate must remain stable TWHGV after the rising $\overline{WR}$ edge for level gating; and for edge gating meaningful transitions (defined above) on the gate must not occur TGVWH before the rising $\overline{WR}$ edge or sooner than TWHEH after the rising $\overline{WR}$ edge. These parameters must be met for all commands issued to an armed counter and for both ARM and LOAD-and-ARM commands issued to a disarmed counter. As above, use the separate LOAD and ARM commands if these parameters can not be met. The 8253 counters are susceptible to erroneous operation if a rising $\overline{WR}$ edge is applied to a counter

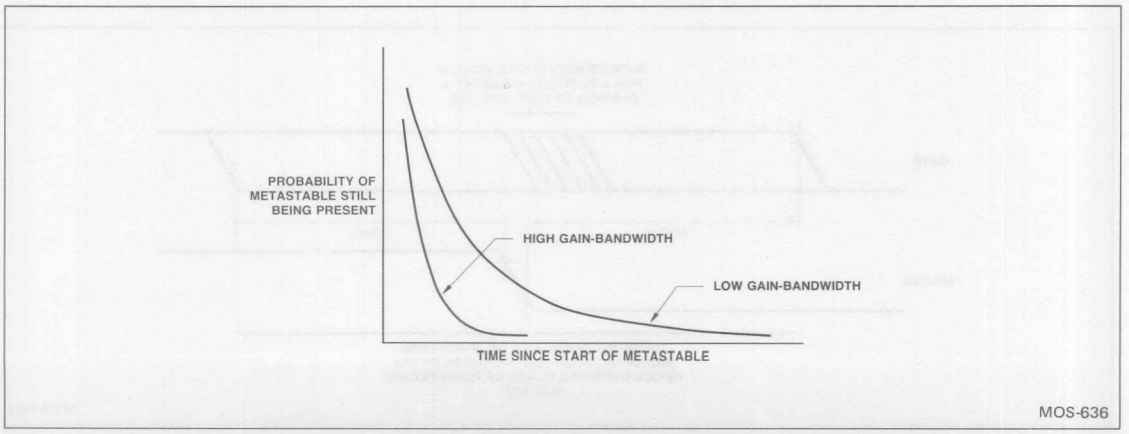


Figure A-1. Metastable Decay Probabilities

SAVE can be applied to more than one counter at a time. For such commands, each counter being acted on must meet parameters TEHWH, TWHEH, TGVWH and TWHGV.

Failure to meet one or more of the Am9513 parameters TEHWH, TNHEH, TGVWH or TWHGV or failure to meet the unspecified 8253 restrictions or when write commands can be issued may result in incorrect execution of the entered command. For example, with a Am9513 SAVE command or an 8253 counter latching operation, an incorrect value might be stored. It is also possible, although likely less probable, that a command entered in violation of these parameters may have a more drastic effect, perhaps altering the counter's contents or perhaps locking the counter up. It is because of the unpredictable nature of metastables that it is not possible to pinpoint the command failure mode that may be experienced. Many users provide reasonableness checking routines in their software to help detect if an error arises, perhaps performing two (Am9513) SAVE or two (8253) counter latching operations, for example, and comparing the results.

With the Am9513, it is sometimes necessary to calculate the relationship between one of the internal F signals (F1 through F5) and some externally applied signal, perhaps a gate or $\overline{WR}$ edge. If FOUT is driven from an internal F signal, say F5, any internal F signal, say F3, can be referred to the FOUT transition using data sheet parameters TFN and TEHFV. In the above example, the maximum delay between a transition on F3 and a transition on FOUT is given by TFN (F3 to F4 delay) + TFN (F4 to F5 delay) + THEFV (FOUT source to FOUT output delay) = 2 TFN + TEHFV. It turns out that the internal F1 signal can be assumed to be equivalent to (i.e., has 0 ns skew relative to) the X2 input. This can be used to reference X2 to one of the internal F signals, for example, to F3. The X2 to F3 maximum delay is simply: (0 ns (X2 to F1 delay) + TFN (F1 to F2 delay) + TFN (F2 to F3 delay)) = 2 TFN. Using the above techniques, any F signal can be referenced to FOUT (if FOUT has one of the F signals as a source) or to X2. Since the relationship between X2 or FOUT to the internal F signal is known, the X2 or FOUT signal may be used to synchronize signals, perhaps an asynchronous gate or $\overline{WR}$ edge, relative to the internal F signal. Note, however, to avoid biasing problems with the Am9513's internal oscillator, if X2 connects to a the Am9513's internal oscillator.

In many systems it may be difficult or impossible to meet some or all of these parameters. In recognition of this, the Am9513 has on-board circuitry designed to synchronize signals violating the above parameters. Because of the low gain-bandwidth product available in MOS technology vis-a-vis bipolar technology, there is a small but significant probability of failure in the Am9513's synchronizing circuitry. Again at the risk of being repetitious, problems because of synchronization failure can only occur if the user does not meet data sheet parameters TGVEH, TEHGV, TEHWH, TWHEH, TGVWH and TWHGV (Am9513) or t_{GS} and t_{GH} (8253).

Calculations can be made to compute the expected failure rate for truly asynchronous signal entry. To calculate the expected metastable error rate, it must be assumed that the two input signals being analyzed are truly asynchronous. Signals which are generated from a common source but which may be skewed relative to each other because of indeterminate propagation delays in their separate paths are not truly asynchronous. The word synchronous, as used in this document, refers to all of these non-asynchronous signals, i.e., to any signals with a predictable timing relationship.

The reason the signal must be truly asynchronous is related to the setup/hold window specified by the parameters in the data sheet. Any given chip will have a setup/hold window much smaller than that given in the data sheet. The typical setup/hold window requirements for a particular counter may actually be many orders of magnitude smaller than the one given in the data sheet. See Figure A-2. This very narrow window for different parts will appear at different points within the data sheet's required setup/hold window. Also, for a given part, this very narrow window will move with changes in operating conditions (voltage, temperature, etc.). The data sheet parameters specify the outer limits of these variations for worst-case conditions.

If signal transitions occur within the narrow setup/hold window of a given part, there is a high probability of a metastable occurring. If the applied signals violating the setup/hold time are truly asynchronous, the probability of their violating the setup/hold window is random. If, on the other hand, the two signals are derived from a common signal or crystal, they in fact will have some predictable

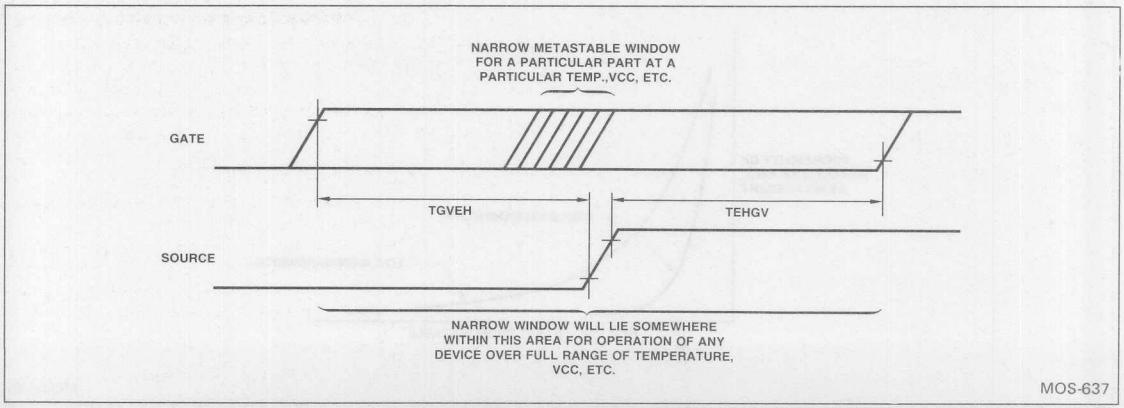


Figure A-2. Am9513 Setup-Hold Requirements

relationship and, at a given set of operating conditions, may fall in the setup/hold window time after time, causing high failure rates.

For this reason, if the Am9513's or 8253's synchronization parameters can not be met, the violating signal pairs must be truly asynchronous to each other in order to set the probability odds on the side of the user. More specifically, if parameters TGVEH or TEHGV (Am9513) or t_{GS} or t_{GH} (8253) can not be met, the gate and source should be asynchronous. If TEHWH or TWHEH (Am9513) can not be met, the $\overline{WR}$ and source signals should be asynchronous (i.e., do not clock the Am9513 off the processor's clock, or vice versa). If TGVWH or TWHGV can not be met, the $\overline{WR}$ and gate signals should be asynchronous. Since the 8253's data sheet does not specify when it is safe to issue write commands the user cannot be certain metastables will not occur.

Assume there is a circuit which violates at least one of the Am9513's or 8253's synchronization parameters and an approximation is desired of the failure rate that may occur. To a first order approximation, one may use the formula

$$P(meta) = r_1\, r_2\, t_w$$

where

P(meta) = probability of a metastable over a given period of time

r_1, r_2 = transition rates of two input signals

t_w = setup/hold window size of a particular part

Note that the equations presented in this appendix are useful for analyzing the probability of metastables, not only for the Am9513 and 8253 but also for any other sequential logic circuit having asynchronous inputs.

Variables r_1 and r_2 are the rate of relevant transitions on the two input signals. For Am9513 sources, only the active-going source edge must be synchronized to the gate or command input. The inactive-going source edge has no effect on the counter. For the 8253, the falling source edge must be synchronized to the gate. For Am9513 command entry, only the rising (trailing) $\overline{WR}$ edge must be synchronized; the falling (leading) $\overline{WR}$ edge can be asynchronous. In Am9513 level gating modes and 8253 modes 0, 2, 3 and 4, both active and inactive gate edges must be synchronized to the source, whereas in Am9513 edge gating applications or 8253 modes 1 and 5, only the meaningful gate edges have significance. (See the definition of meaningful gate edges earlier in this appendix.) Thus for all inputs except Am9513 level gate signals or 8253 gate signals in modes 0, 2, 3 and 4, parameters r_1 or r_2 are equal to the frequencies of the inputs. For level gating, since both edges have significance, the transition rate r_1 or r_2 is twice the gate signal frequency.

Variable t_w is the narrow setup/hold window for a given part under a certain set of operating conditions (see Figure A-2). The actual value of this variable will be highly dependent on the application, and for this reason it is best determined empirically for the particular application being studied.

One approach useful in determining P(meta) is to run the application at a much higher rate than is normally intended. This should generate an easily measurable error rate. The value of t_w can then be calculated and used to calculate P(meta) for the normal rate of operation.

For example, suppose a user plans to asynchronously level-gate at a rate of 10Hz while using a 1000Hz source. The user wishes to calculate P(meta) for this application. Parameter t_w might be calculated while running a 2MHz signal into the source while level-gating at 200kHz. He might observe the counter output to verify correct operation. By observing the error rate over a period of time, the above formula could be used to calculate t_w, as follows:

$$t_w = \frac{P(meta)'}{r_1'\, r_2'}$$

Note in our application $r_1' = 2 \times 10^6$ and $r_2' = 400 \times 10^3$. Parameter r_2' is twice the gate frequency since both gate edges must be synchronized. The empirically determined t_w can now be used to find P(meta) for the intended application using the formula

$$P(meta) = r_1\, r_2\, t_w$$

with r_1 = 1000Hz and r_2 = 20Hz, corresponding to the source and gate repetition rates for the desired application.

In many cases, no errors will be observable even at accelerated operating rates. The ratio of the accelerated rate to the final applications rate allows extrapolation of the period of time likely between errors. For example, if our user observed the accelerated application for ten minutes and no error was seen, he could extrapolate the time between errors for the final application as follows:

$$P(meta) = \frac{r_1\, r_2\, P(meta)'}{r_1'\, r_2'} = \frac{1000\ Hz \times 20\ Hz \times \frac{1\ error}{10\ minutes}}{2 \times 10^6\ Hz \times 400 \times 10^3\ Hz}$$

$$= 2.5 \times 10^{-9}\ errors/sec = 1\ error/4 \times 10^8\ minutes$$

Thus, the final application's average error rate will likely be better than once every 760 years.

In some applications, more than one Am9513 setup/hold parameter may be violated. (Since the 8253 data sheet only specifies the source-gate setup requirements, the user can not be sure whether applied write commands are in violation of the setup and hold requirements of the 8253's gate and source.) For each setup/hold window possibly violated (TGVEH and TEHGV; TEHWH and TWHEH; and TGVWH and TWHGV), the related pairs of signals (gate-source; source – $\overline{WR}$; and gate – $\overline{WR}$) should be run at accelerated rates to compute t_w'.

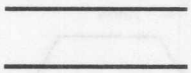
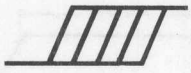
The formula can then be used to estimate P(meta) for each of the setup/hold requirements. The resultant error probability is then the sum of the P(meta) values for each setup/hold window.

For example, our above discussion calculated P(meta) for the TGVEH – TEHGV source-gate setup/hold window. We might also calculate P(meta) for the TEHWH – TWHEH source – $\overline{WR}$ window by using accelerated source and WR signals, keeping the gate repetition rate at the final application rate of 10Hz to minimize its effect. If our new source – $\overline{WR}$ P(meta) was 1 error every 500 years, the combined average error rate would be (1/500 + 1/760) or 1 error every 300 years. If the gate – $\overline{WR}$ setup/hold parameters were also violated, the P(meta) for this window should also be calculated as a component in the final error rate. As mentioned earlier, these calculations can also be used to analyze metastable error rates for any other sequential logic circuit having asynchronous inputs.

Three cautionary notes are appropriate at this point. First, the formula provides a first order approximation only and should be used only for rough estimates. Second, since metastable errors will occur at random times for asynchronous signals, multiple measurements and use of statistical techniques should be used to minimize the effect of random fluctuations in the measured data. Third, the desired application should be thoroughly tested at the normal operating frequencies. This approximation technique of estimating error rates should be used to further substantiate the results of this through testing rather than be used in lieu of thoroughly testing the application.

APPENDIX B – KEY TO TIMING DIAGRAMS

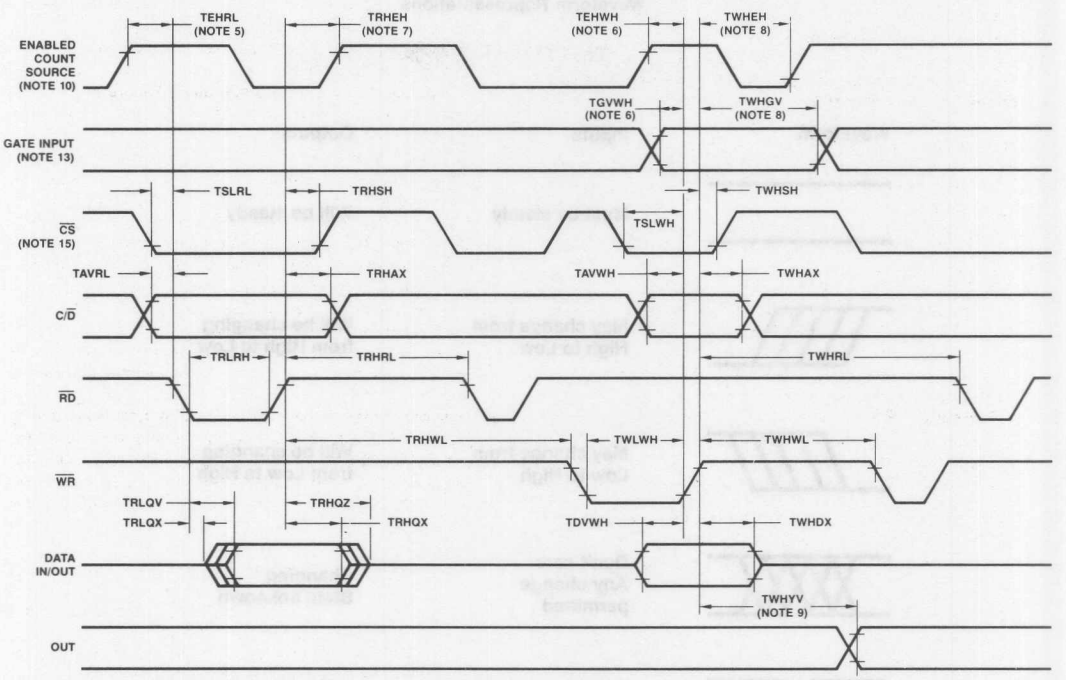
Waveform Representations

Waveform	Inputs	Outputs
	Must be steady	Will be steady
	May change from High to Low	Will be changing from High to Low
	May change from Low to High	Will be changing from Low to High
	Don't care Any change permitted	Changing State unknown
	May make one change from High to Low or Low to High	Will make one change from High to Low or Low to High

MOS-638

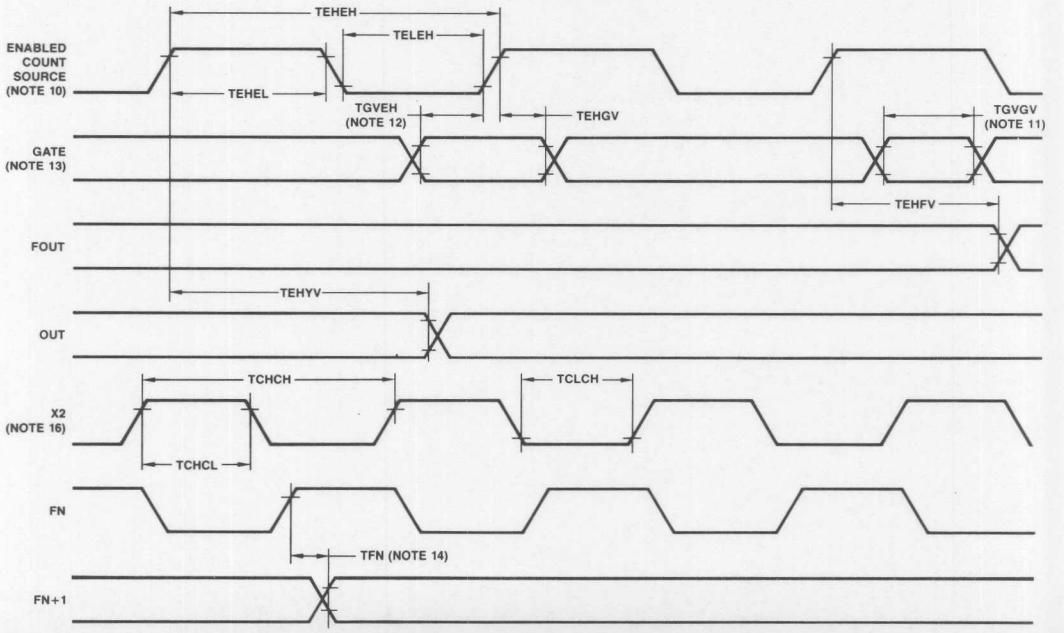
APPENDIX B – Am9513 SWITCHING WAVEFORMS

Bus Transfer Switching Waveforms



MOS-183

Counter Switching Waveforms



MOS-184

APPENDIX B – Am9513 SWITCHING WAVEFORMS

NOTES:

1. Typical values are for T_A = 25°C, nominal supply voltage and nominal processing parameters.
2. Test conditions assume transition times of 10ns or less, timing reference levels of 0.8V and 2.0V and output loading of one TTL gate plus 100pF, unless otherwise noted.
3. Abbreviations used for the switching parameter symbols are given as the letter T followed by four or five characters. The first and third characters represent the signal names on which the measurements start and end. Signal abbreviations used are:

   A (Address) = C/$\overline{D}$
   C (Clock) = X2
   D (Data In) = DB0-DB15
   E (Enabled counter source input) = SRC1-SRC5, GATE1-GATE5, F1-F5, TCN–1
   F = FOUT
   G (Counter gate input) = GATE1-GATE5, TCN–1
   Q (Data Out) = DB0-DB15
   R (Read) = $\overline{RD}$
   S (Chip Select) = $\overline{CS}$
   W (Write) = $\overline{WR}$
   Y (Output) = OUT1-OUT5

   The second and fourth letters designate the reference states of the signals named in the first and third letters respectively, using the following abbreviations.

   H = High
   L = Low
   V = Valid
   X = unknown or don't care
   Z = high impedance

4. Switching parameters are listed in alphabetical order.
5. Any input transition that occurs before this minimum setup requirement will be reflected in the contents read from the status register.
6. Any input transition that occurs before this minimum setup requirement will act on the counter before the execution of the operation initiated by the write. Failure to meet this setup time when issuing commands to the counter may result in incorrect counter operation.
7. Any input transition that occurs after this minimum hold time is guaranteed to not influence the contents read from the status register on the current read operation.
8. Any input transition that occurs after this minimum hold time is guaranteed to be seen by the counter as occurring after the action initiated by the write operation. Failure to meet this hold time when issuing commands to the counter may result in incorrect counter operation.
9. This parameter applies to cases where the write operation causes a change in the output bit.
10. The enabled count source is one of F1-F5, TCN–1, SRC1-SRC5 or GATE1-GATE5, as selected in the applicable Counter Mode register. The timing diagram assumes the counter counts on rising source edges. The timing specifications are the same for falling-edge counting.
11. This parameter applies to edge gating (CM15-CM13 = 110 or 111) and gating when both CM7 = 1 and CM15-CM13 ≠ 000. This parameter represents the minimum GATE pulse width needed to ensure that the pulse initiates counting or counter reloading.
12. This parameter applies to both edge and level gating (CM15-CM13 = 001 through 111) and gating when both CM7 = 1 and CM15-CM13 = 000. This parameter represents the minimum setup or hold times to ensure that the Gate input is seen at the intended level on the active source edge. Failure to met the required setup and hold times may result in incorrect counter operation.
13. This parameter assumes that the GATENA input is unused (16-bit bus mode) or is tied high. In cases where the GATENA input is used, this timing specification must be met by both the GATE and GATENA inputs.
14. Signals F1-F5 cannot be directly monitored by the user. The phase difference between these signals will manifest itself by causing counters using two different F signals to count at different times on nominally simultaneous transitions in the F signals.
15. This timing specification assumes that $\overline{CS}$ is active whenever $\overline{RD}$ or $\overline{WR}$ are active. $\overline{CS}$ may be held active indefinitely.
16. This parameter assumes X2 is driven from an external gate with a square wave.
17. This parameter assumes that the write operation is to the command register.

The following Am9513 data model summary collects together all C structures appearing in the text of this note for reference purposes.

```
/*
        Record definitions - register fields for Am9513 data model
*/
                                    /* Master mode Register */

typedef RECORD  {unsigned day_mode       : 2 ;
                 unsigned compar_1       : 1 ;
                 unsigned compar_2       : 1 ;
                 unsigned FOUT_source    : 4 ;
                 unsigned FOUT_divisor   : 4 ;
                 unsigned FOUT_gate      : 1 ;
                 unsigned data_bus       : 1 ;
                 unsigned data_ptr       : 1 ;
                 unsigned scaler         : 1 ;
                } master_type ;

                                    /* Counter Mode Register */

typedef RECORD  {unsigned output         : 3 ;
                 unsigned direction      : 1 ;
                 unsigned base           : 1 ;
                 unsigned control        : 3 ;
                 unsigned source         : 4 ;
                 unsigned edge           : 1 ;
                 unsigned gate           : 3 ;
                } count_type ;

                                    /* Status Register      */

typedef RECORD  {unsigned byte_ptr       : 1 ;
                 unsigned output1        : 1 ;
                 unsigned output2        : 1 ;
                 unsigned output3        : 1 ;
                 unsigned output4        : 1 ;
                 unsigned output5        : 1 ;
                 unsigned not_used       : 2 ;
                } status_type ;

                                    /* Load data ptr register command */

typedef RECORD  {unsigned group          : 3 ;
                 unsigned element        : 2 ;
                 unsigned cmnd_code      : 3 ;
                } data_type ;

/*      - chip level structures */

                                    /* Single counter set   */

typedef RECORD  {count_type      mode ;
                 unsigned int    load ;
                 unsigned int    hold ;
                } channel_type ;

                                    /* Am9513 chip set      */

typedef RECORD  {master_type     master ;
                 channel_type    counter [5] ;
                 status_type     status ;
                 unsigned int    alarm_1 ;
                 unsigned int    alarm_2 ;
                } AM9513_type ;
```

This appendix defines the macro command available with detailed syntax of use. The macros are available for the following assemblers:

1. Am8080/Am8085 MACRO8 assembler (AMD)
2. Z80 RIO assembler (Zilog)
3. Z8000 MACZ assembler (AMD)

The available macros are summarised here with details of usage. Notice that in the following text angle brackets ("<>") are used to describe parameters to be entered and as such are not actually to be entered. Square brackets ("[]") describe optional items.

The "OR" symbol ("|") signifies that only one of the parameters given should be entered.

The comma is a legal parameter separator for all three macro files.

RESET	
FOUT	ON|OF
DPS	ON|OF
POINT	<group>,<element>
ARM	<combination>
LOAD	<combination>
LD_ARM	<combination>
SAVE	<combination>
DRMSAV	<combination>
SET_	<counter>
CLEAR	<counter>
STEP	<counter>
LOAD_REG	<counter>,<constant>[,<indirection>]
HOLD_REG	<counter>,<constant>[,<indirection>]
MODE_REG	<counter>,<output>,<direction>,<base>,<control>,<source>,<edge>,<gate>
MASTER	<day_mode>,<compar_1>,<compar_2>,<FOUT_source>,<FOUT_divisor>,<FOUT_gate>,<data_bus>,<data_ptr>,<scaler>

```
<combination>    ::=    <counter>|<counter>,<combination>

<counter>        ::=    1|2|3|4|5

<indirection>    ::=    I

<group>          ::=    <counter>|CTRL_GR

<element>        ::=    MODE_|LOAD_|HOLD_|HOLD_CY
                        |ALARM1_|ALARM2_|MASTER_|STATUS_

<output>         ::=    OF_LO_TC|ACT_HI_TC|TC_TOGGLE|OF_OC_TC|ACT_LO_TC

<direction>      ::=    UP|DOWN

<base>           ::=    <scaler>

<control>        ::=    <allowed_modes>

<source>         ::=    <FOUT_source>|TC_NM1

<edge>           ::=    RISE|FALL

<gate>           ::=    NO_GATE|HL_TC_NM1|HL_NP1_GATE|HL_NM1_GATE|HL_GATE_N
                        |LL_GATE_N|HE_GATE_N|LE_GATE_N

<day_mode>       ::=    TOD_OFF|TOD_50HZ|TOD_60HZ|TOD_100HZ


<compar_2>       ::=    <compar_1>

<compar_1>       ::=    ENABLE|DISABLE

<FOUT_source>    ::=    SRC_1|SRC_2|SRC_3|SRC_4|SRC_5
                        |GATE_1|GATE_2|GATE_3|GATE_4|GATE_5
                        |F1|F2|F3|F4|F5

<FOUT_divisor>   ::=    <hex_digit>

<FOUT_gate>      ::=    ON|OF

<data_bus>       ::=    BUS_8|BUS_16

<data_ptr>       ::=    ON|OF

<scaler>         ::=    BINARY|BCD

<hex_digit>      ::=    <digit>|0AH|0BH|0CH|0DH|0EH|0FH

<constant>       ::=    <digit>|<digit><constant>

<digit>          ::=    0|1|2|3|4|5|6|7|8|9

<allowed_modes>  ::=    MODE_ABC|MODE_DEF|MODE_GHI|MODE_JKL|MODE_mNO
                        |MODE_pQR|MODE_Stu|MODE_Vwx
```

The following macro code definitions implement the macro commands as listed in the Macro Command Summary (Appendix C). Notice that all macros use only the A and F registers, all other working registers are unaffected. These macros are targeted for the AMD MACRO8 assembler.

```
;       Am9513 Macro definitions for Am8080 Macro8 Assembler

S_MASK  MACRO   P1,P2,P3,P4,P5

        ;; Recursive macro to get correct s-field

        IFNB    <P2>
        S_MASK  P2,P3,P4,P5
        ENDIF
        IFT     (0 LT P1) AND (P1 LT 6)
DLAB    SET     DLAB OR (1 SHI (P1-1))
        ELSE
        ??? P1  ; Illegal Counter
        ENDIF
        ENDM

S_TYPE  MACRO   X,P1,P2,P3,P4,P5

        ;; Optimised sequence for multiple register s-field commands

        IFT     X EQ 20H OR X EQ 40H OR X EQ 60H OR X EQ 202H OR X EQ 2A2H
DLAB    SET     X                                                  OR X EQ 80H
        S_MASK  P1,P2,P3,P4,P5
        MVI     A,DLAB
        OUT     A_CTRL
        ELSE
        ??? X   ; Illegal s-mask command code
        ENDIF
        ENDM

N_TYPE  MACRO   P1,P2

        ;; Optimised sequence for single register n-field commands

        IFT     (0 LT P1) AND (P1 LT 6)
        MVI     A,P1 OR P2
        OUT     A_CTRL
        ELSE
        ??? P1  ; Illegal Counter
        ENDIF
        ENDM

ARM     MACRO   P1,P2,P3,P4,P5          ;; Arm counters, any from 1,5
        S_TYPE  20H,P1,P2,P3,P4,P5
        ENDM

LOAD    MACRO   P1,P2,P3,P4,P5          ;; Load counters, any from 1,5
        S_TYPE  40H,P1,P2,P3,P4,P5
        ENDM

LD_ARM  MACRO   P1,P2,P3,P4,P5          ;; Load 'n Arm counters, any from 1,5
        S_TYPE  60H,P1,P2,P3,P4,P5
        ENDM
```

```
DISARM  MACRO   P1,P2,P3,P4,P5          ;; Disarm counters, any from 1,5
        S_TYPE  0C0H,P1,P2,P3,P4,P5
        ENDM

SAVE    MACRO   P1,P2,P3,P4,P5          ;; Save counters, any from 1,5
        S_TYPE  0A0H,P1,P2,P3,P4,P5
        ENDM

DRMSAV  MACRO   P1,P2,P3,P4,P5          ;; Disarm 'n Save counters. any from 1,5
        S_TYPE  80H,P1,P2,P3,P4,P5
        ENDM

SET_    MACRO   P1                      ;; Set single counter output
        N_TYPE  P1,0E8H
        ENDM

CLEAR   MACRO   P1                      ;; Clear single counter output
        N_TYPE  P1,0E0H
        ENDM

STEP    MACRO   P1                      ;; Step single counter output
        N_TYPE  P1,0F0H
        ENDM

FOUT    MACRO   P1                      ;; Gate FOUT on or off
        IFT     '&P1' EQ 'OF'
        MVI     A,0EEH
        OUT     A_CTRL
        ELSE
        IFT     '&P1' EQ 'ON'
        MVI     A,0E6H
        OUT     A_CTRL
        ELSE
        ??? P1  ; Illegal request
        ENDIF
        ENDIF
        ENDM

RESET   MACRO                           ;; Reset, load all counters
        MVI     A,0FFH
        OUT     A_CTRL
        LOAD    1,2,3,4,5
        ENDM

POINT   MACRO   P1,P2           ;; Set data pointer register group, element

        IFT     ((0 LT P1 AND P1 LT 6) OR P1 EQ 7) AND 0 LE P2 AND P2 LT 4
        MVI     A,P1 OR (P2 SHL 3)
        OUT     A_CTRL
        ELSE
        ??? P1,P2       ; Illegal request
        ENDIF
        ENDM
```

```
DPS     MACRO   P1              ;; Set data pointer sequencing on/of
        IFT     '&P1' EQ 'ON'
        MVI     A,0E0H
        OUT     A_CTRL
        ELSE
        IFT     '&P1' EQ 'OF'
        MVI     A,0E8H
        OUT     A_CTRL
        ELSE
        ??? P1  ; Illegal request
        ENDIF
        ENDIF
        ENDM


MASTER  MACRO   P1,P2,P3,P4,P5,P6,P7,P8,P9
                                        ;; Set Master register
DLAB    SET     P1                      ;;  .day_mode
DLAB    SET     DLAB OR (P2 SHL 2)      ;;  .compar_1
DLAB    SET     DLAB OR (P3 SHL 3)      ;;  .compar_2
DLAB    SET     DLAB OR (P4 SHL 4)      ;;  .FOUT_source
DLAB    SET     DLAB OR (P5 SHL 8)      ;;  .FOUT_divisor
DLAB    SET     DLAB OR (P6 SHL 12)     ;;  .FOUT_gate
DLAB    SET     DLAB OR (0 SHL 13)      ;;  .data_bus - 8 bit only
DLAB    SET     DLAB OR (P8 SHL 14)     ;;  .data_ptr
DLAB    SET     DLAB OR (P9 SHL 15)     ;;  .scaler
        POINT   CTRL_GR,MASTER_         ;; Point to Master Register
        MVI     A, LOW DLAB
        OUT     A_DATA                  ;; Send Low byte first
        MVI     A, HIGH DLAB
        OUT     A_DATA                  ;; Then High byte
        ENDM

MODE_REG MACRO  P0,P1,P2,P3,P4,P5,P6,P7
                                        ;; Set counter P0 Mode register
        IFT     P1 EQ 3 OR 5 LT P1
        ??? P1  ; Illegal Output Control
        ELSE
DLAB    SET     P1                      ;;  .output
DLAB    SET     DLAB OR (P2 SHL 3)      ;;  .direction
DLAB    SET     DLAB OR (P3 SHL 4)      ;;  .base
DLAB    SET     DLAB OR (P4 SHL 5)      ;;  .control
DLAB    SET     DLAB OR (P5 SHL 8)      ;;  .source
DLAB    SET     DLAB OR (P6 SHL 12)     ;;  .edge
DLAB    SET     DLAB OR (P7 SHL 13)     ;;  .gate
        IFT     0 LT P0 AND P0 LT 6
        POINT   P0,MODE_                ;; Point to counter P1 Mode
        MVI     A, LOW DLAB
        OUT     A_DATA                  ;; Send Low byte
        MVI     A, HIGH DLAB
        OUT     A_DATA                  ;; Then High byte
        ELSE
        ??? P0  ; Illegal counter #
        ENDIF
        ENDIF
        ENDM
```

```
LOAD_REG MACRO  P0,P1,P2        ;; Set Load reg of counter P0 to P1
         IFNB   <P2>            ;; Test for indirection
         POINT  P0,LOAD_        ;; Nested macro must lie within the test
         LDA    P1
         OUT    A_DATA          ;; Low byte
         LDA    P1+1
         OUT    A_DATA          ;; High byte
         ELSE
         POINT  P0,LOAD_
         MVI    A, LOW P1
         OUT    A_DATA          ;; Low byte
         MVI    A, HIGH P1
         OUT    A_DATA          ;; High byte
         ENDIF
         ENDM

HOLD_REG MACRO  P0,P1,P2        ;; Set Hold Reg of counter P0 to P1
         IFNB   <P2>            ;; Test for indirection
         POINT  P0,HOLD_        ;; Nested macro must lie within the test
         LDA    P1
         OUT    A_DATA          ;; Low byte
         LDA    P1+1
         OUT    A_DATA          ;; High byte
         ELSE
         POINT  P0,HOLD_
         MVI    A, LOW P1
         OUT    A_DATA          ;; Low byte
         MVI    A, HIGH P1
         OUT    A_DATA          ;; High byte
         ENDIF
         ENDM



A>
```

The following macro code definitions implement the macro command as listed in the Macro Command Summary (Appendix D). Notice that all macros use only the A and F registers, all other working registers are unaffected. These macros are targeted for the Zilog RIO assembler.

```
S_MASK  MACRO   #1 #2 #3 #4 #5

        ; Recursive macro to get correct s-field

        COND    '#2'
        S_MASK  #2 #3 #4 #5
        ENDC
        COND    (0<#1)&(#1<6)
DLAB    DEFL    DLAB.OR.(1.SHL.(#1-1))
        ENDC
        COND    .NOT.((0<#1)&(#1<6))
        ??? #1  ; Illegal counter #
        ENDC
        ENDM

S_TYPE  MACRO   #0 #1 #2 #3 #4 #5

        ; Optimised sequence for multiple register s-field commands

        COND    (#0=20H)^(#0=40H)^(#0=60H)^(#0=0C0H)^(#0=0A0H)^(#0=80H)
DLAB    DEFL    #0
        S_MASK  #1 #2 #3 #4 #5
        LD      A,DLAB
        OUT     (A_CTRL),A
        ENDC
        COND    .NOT.((#0=20H)^(#0=40H)^(#0=60H)^(#0=0C0H)^(#0=0A0H)^(#0=80H))
        ??? #0  ; Illegal command
        ENDC
        ENDM

N_TYPE  MACRO   #1 #2

        ; Optimised sequence for single register n-type commands

        COND    (0<#1)&(#1<6)
        LD      A,#1.OR.#2
        OUT     (A_CTRL),A
        ENDC
        COND    .NOT.((0<#1)&(#1<6))
        ??? #1  ; Illegal counter
        ENDC
        ENDM

ARM     MACRO   #1 #2 #3 #4 #5  ; Arm counters, any from 1,5
        S_TYPE  20H #1 #2 #3 #4 #5
        ENDM

LOAD    MACRO   #1 #2 #3 #4 #5  ; Load counters, any from 1,5
        S_TYPE  40H #1 #2 #3 #4 #5
        ENDM

LD_ARM  MACRO   #1 #2 #3 #4 #5  ; Load 'n Arm counters, any from 1,5
        S_TYPE  60H #1 #2 #3 #4 #5
        ENDM
```

```
        S_TYPE  0C0H #1 #2 #3 #4 #5
        ENDM

SAVE    MACRO   #1 #2 #3 #4 #5  ; Save counters, any from 1,5
        S_TYPE  0A0H #1 #2 #3 #4 #5
        ENDM

DRMSAV  MACRO   #1 #2 #3 #4 #5  ; Disarm 'n Save counters, any from 1,5
        S_TYPE  80H #1 #2 #3 #4 #5
        ENDM

SET_    MACRO   #1              ; Set single counter output #1
        N_TYPE  #1,0E8H
        ENDM

CLEAR   MACRO   #1              ; Clear single counter output #1
        N_TYPE  #1,0E0H
        ENDM

STEP    MACRO   #1              ; Step single counter output #1
        N_TYPE  #1,0F0H
        ENDM

FCUT    MACRO   #1              ; Gate FCUT on or off
        COND    '#1'='OF'
        LD      A,0EEH
        CUT     (A_CTRL),A
        ENDC
        COND    '#1'='ON'
        LD      A,0E6H
        OUT     (A_CTRL),A
        ENDC
        COND    .NOT.(('#1'='OF').OR.('#1'='ON'))
        ??? #1  ; Illegal option
        ENDC
        ENDM

RESET   MACRO                   ; Reset, load all counters
        LD      A,0FFH
        CUT     (A_CTRL),A
        LOAD    1,2,3,4,5
        ENDM

POINT   MACRO   #1 #2   ; Set data pointer register group, element

        COND    ((0<#1)&(#1<6)).OR.((#1=7)&((0.IE.#2)&(#2<4)))
        LD      A #1.OR.(#2.SHL.3)
        CUT     (A_CTRL),A
        ENDC
        COND    .NOT.(((0<#1)&(#1<6)).OR.((#1=7)&((0.IE.#2)&(#2<4))))
        ??? #1 #2
        ENDC
        ENDM
```

```
DPS     MACRO   #1      ; Set data pointer sequencing on/of
        COND    '#1'='ON'
        LD      A,0E0H
        OUT     (A_CTRL),A
        ENDC
        COND    '#1'='OF'
        ID      A,0E8H
        OUT     (A_CTRL),A
        ENDC
        COND    .NOT.(('#1'='OF').OR.('#1'='ON'))
        ??? #1  ; Illegal option
        ENDC
        ENDM

MASTER  MACRO   #1 #2 #3 #4 #5 #6 #7 #8 #9
                                        ;  Set Master register
DLAB    DEFL    #1                      ;  .day_mode
DLAB    DEFL    DLAB.OR.(#2.SHL.2)      ;  .compar_1
DLAB    DEFL    DLAB.OR.(#3.SHL.3)      ;  .compar_2
DLAB    DEFL    DLAB.OR.(#4.SHL.4)      ;  .FOUT_source
DLAB    DEFL    DLAB.OR.(#5.SHL.8)      ;  .FOUT_divisor
DLAB    DEFL    DLAB.OR.(#6.SHL.12)     ;  .FOUT_gate
DLAB    DEFL    DLAB.OR.(0.SHL.13)      ;  .data_bus - 8 bit only
DLAB    DEFL    DLAB.OR.(#8.SHL.14)     ;  .data_ptr
DLAB    DEFL    DLAB.OR.(#9.SHL.15)     ;  .scaler
        POINT   CTRL_GR,MASTER_         ; Point to Master Register
        LD      A,DLAB.MOD.256
        OUT     (A_DATA),A              ; Send Low byte first
        LD      A,DLAB/256
        OUT     (A_DATA),A              ; Then High byte
        ENDM

MODE    MACRO   #0 #1 #2 #3 #4 #5 #6 #7
                                        ;  Set counter #0 mode register
        COND    (#1=3).OR.(5<#1)
        ??? #1  ; Illegal Output Control
        ENDC
        COND    .NOT.((#1=3).OR.(5<#1))
DLAB    DEFL    #1                      ;  .output
DLAB    DEFL    DLAB.OR.(#2.SHL.3)      ;  .direction
DLAB    DEFL    DLAB.OR.(#3.SHL.4)      ;  .base
DLAB    DEFL    DLAB.OR.(#4.SHL.5)      ;  .control
DLAB    DEFL    DLAB.OR.(#5.SHL.8)      ;  .source
DLAB    DEFL    DLAB.OR.(#6.SHL.12)     ;  .edge
DLAB    DEFL    DLAB.OR.(#7.SHL.13)     ;  .gate
        COND    (0<#0)&(#0<6)
        POINT   #0,MODE_                ; Point to counter #1 Mode
        LD      A,DLAB.MOD.256
        OUT     (A_DATA),A              ; Send Low byte
        LD      A,DLAB/256
        OUT     (A_DATA),A              ; Then High byte
        ENDC
        COND    .NOT.((0<#0)&(#0<6))
        ??? #0  ; Illegal counter #
        ENDC
        ENDM
```

```
LOAD_REG MACRO  #0 #1 #2        ; Set Load reg of counter #0 to #1
         POINT  #0,LOAD_
         COND   '#2'            ; Test for indirection
         LD     A,(#1)
         OUT    (A_DATA),A      ; Low byte
         LD     A,(#1+1)
         OUT    (A_DATA),A      ; High byte
         ENDC
         COND   .NOT.('#2')
         LD     A,#1.MOD.256
         OUT    (A_DATA),A      ; Low byte
         LD     A,#1/256
         OUT    (A_DATA),A      ; High byte
         ENDC
         ENDM

HOLD_REG MACRO  #0 #1 #2        ; Set Hold Reg of counter #0 to #1
         POINT  #0,HOLD_
         COND   '#2'            ; Test for indirection
         LD     A,(#1)
         OUT    (A_DATA),A      ; Low byte
         LD     A,(#1+1)
         OUT    (A_DATA),A      ; High byte
         ENDC
         COND   .NOT.('#2')
         LD     A,#1.MOD.256
         OUT    (A_DATA),A      ; Low byte
         LD     A,#1/256
         OUT    (A_DATA),A      ; High byte
         ENDC
         ENDM


A>
```

The following macro code definitions implement the macro commands as listed in the Macro Command Summary (Appendix D). Notice that all macros use only registers RØ. The flag and control word register is altered but all other working registers are unaffected. These macros are targeted for the AMD MACZ assembler.

```
VAR     DLAB: OBJECT;                       % Define a dummy variable
        DLAB    ::= 0;

MACRO   S_MASK  AA;                         % Build up a dummy s_field
        BEGIN
        IF 0 LT AA AND AA LT 6
        THEN
                DLAB ::= DLAB OR (1 SHL (AA-1))
        ELSE
                CALR 0FFFFH;                % Invalid counter #
        END;

MACRO   S_TYPE  BB,BB1,BB2,BB3,BB4,BB5;

        % Optimised sequence for multiple register s-field commands

        BEGIN
        DLAB    ::= BB OR 0FF00H;
        IF NOT NULL BB5                     % Test for the parameters
        THEN
                S_MASK BB5;
        IF NOT NULL BB4
        THEN
                S_MASK BB4;
        IF NOT NULL BB3
        THEN
                S_MASK BB3;
        IF NOT NULL BB2
        THEN
                S_MASK BB2;
        IF NOT NULL BB1
        THEN
                S_MASK BB1;
        LD      R0,DLAB;
        OUT     A_CTRL,R0                   % Send the command
        END;

MACRO   N_TYPE  CC1,CC2;

        % Optimised sequence for single register n-field commands

        BEGIN
        IF (0 LT CC1) AND (CC1 LT 6)
        THEN
        BEGIN
                LD      R0,CC1 OR CC2 OR 0FF00H;
                OUT     A_CTRL,R0
        END
        ELSE
                CALR 0FFFFH;                % Illegal Counter
        END;
```

```
MACRO   ARM     DD1,DD2,DD3,DD4,DD5;    % Arm counters, any from 1,5
        BEGIN
                S_TYPE  20H,DD1,DD2,DD3,DD4,DD5
        END;

MACRO   LOAD    EE1,EE2,EE3,EE4,EE5;    % Load counters, any from 1,5
        BEGIN
                S_TYPE  40H,EE1,EE2,EE3,EE4,EE5
        END;

MACRO   LD_ARM  FF1,FF2,FF3,FF4,FF5;    % Load 'n Arm counters, any from 1,5
        BEGIN
                S_TYPE  60H,FF1,FF2,FF3,FF4,FF5
        END;

MACRO   DISARM  GG1,GG2,GG3,GG4,GG5;    % Disarm counters, any from 1,5
        BEGIN
                S_TYPE  0C0H,GG1,GG2,GG3,GG4,GG5
        END;

MACRO   SAVE    HH1,HH2,HH3,HH4,HH5;    % Save counters, any from 1,5
        BEGIN
                S_TYPE  0A0H,HH1,HH2,HH3,HH4,HH5
        END;

MACRO   DRMSAV  II1,II2,II3,II4,II5;    % Disarm 'n Save counters, any from 1,5
        BEGIN
                S_TYPE  80H,II1,II2,II3,II4,II5
        END;

MACRO   SET     JJ1;                    % Set single counter output JJ1
        BEGIN
                N_TYPE  JJ1,0E8H
        END;

MACRO   CLEAR   JJ2;                    % Clear single counter output JJ2
        BEGIN
                N_TYPE  JJ2,0E0H
        END;

MACRO   STEP    JJ3;                    % Step single counter output JJ3
        BEGIN
                N_TYPE  JJ3,0F0H
        END;

MACRO   FOUT    JJ4;                    % Gate FOUT on or off
        BEGIN
        IF JJ4 EQ OF
        THEN
        BEGIN
                LD      R0,0FFEEH;
                OUT     A_CTRL,R0
        END
        ELSE
                IF JJ4 EQ ON
                THEN
                BEGIN
                        ID      R0,0FFE6H;
                        OUT     A_CTRL,R0
                END
                ELSE
                        CALR 0FFFFH;    % Illegal request
        END;
```

```
MACRO   RESET   ;                       % Reset, load all counters
        BEGIN
        LD      R0,0FFFFH;
        OUT     A_CTRL,R0;              % Official reset
        LOAD    1,2,3,4,5;              % Clear any set TC's;
        LD      R0,1;
        OUT     A_CTRL,R0;              % Dummy set data pointer
        LD      R0,0FFEFH;
        OUT     A_CTRL,R0               % Set 16 bit data bus
        END;

MACRO   POINT   KK1,KK2 ;               % Set data pointer register group, element

        BEGIN
        IF ((0 LT KK1 AND KK1 LT 6) OR KK1 EQ 7) AND 0 LE KK2 AND KK2 LT 4
        THEN
        BEGIN
                LD      R0,KK1 OR (KK2 SHL 3) OR 0FF00H;
                OUT     A_CTRL,R0
        END
        ELSE
                CALR 0FFFFH;            % Illegal request
        END;

MACRO   DPS     P1;     % Set data pointer sequencing on/of
        BEGIN
        IF P1 EQ ON
        THEN
        BEGIN
                LD      R0,0FFE0H;
                OUT     A_CTRL,R0
        END
        ELSE
                IF P1 EQ OF
                THEN
                BEGIN
                        LD      R0,0FFE8H;
                        OUT     A_CTRL,R0
                END
                ELSE
                        CALR 0FFFFH;    % Illegal request
        END;

MACRO   MASTER  LL1,LL2,LL3,LL4,LL5,LL6,LL7,LL8,LL9;
        BEGIN
        DLAB ::= LL1;                   %  .day_mode
        DLAB ::= DLAB OR (LL2 SHL 2);   %  .compar_1
        DLAB ::= DLAB OR (LL3 SHL 3);   %  .compar_2
        DLAB ::= DLAB OR (LL4 SHL 4);   %  .FOUT_source
        DLAB ::= DLAB OR (LL5 SHL 8);   %  .FOUT_divisor
        DLAB ::= DLAB OR (LL6 SHL 12);  %  .FOUT_gate
        DLAB ::= DLAB OR (1 SHL 13);    %  .data_bus - 16 bit only
        DLAB ::= DLAB OR (LL8 SHL 14);  %  .data_ptr
        DLAB ::= DLAB OR (LL9 SHL 15);  %  .scaler
        POINT   7,2;                    % Point to Master Register
        LD      R0,DLAB;
        OUT     A_DATA,R0               % Then High byte
        END;
```

```
        IF MM1 EQ 3 OR 5 LT MM1
        THEN
                CALR ØFFFFE     % Illegal Output Control
        ELSE
        BEGIN
        DLAB ::= MM1;                           %  .output
        DLAB ::= DLAB OR (MM2 SHL 3);           %  .direction
        DLAB ::= DIAB OR (MM3 SHL 4);           %  .base
        DLAB ::= DLAB OR (MM4 SHL 5);           %  .control
        DLAB ::= DLAB OR (MM5 SHL 8);           %  .source
        DLAB ::= DLAB OR (MM6 SHL 12);          %  .edge
        DLAB ::= DLAB OR (MM7 SHL 13)           %  .gate
        END;
        IF 0 LT MMØ AND MMØ LT 6
        THEN
        BEGIN
                POINT   MMØ,0;                  % Point to counter MM1 Mode
                LD      RØ,DLAB;
                OUT     A_DATA,RØ               % Send 16 bit word
        END
        ELSE
        CALR ØFFFFH;    % Illegal counter
        END;

MACRO   LOAD_REG NNØ,NN1;                       % Set Load reg of NNØ to NN1
        BEGIN
        POINT   NNØ,1;                          % Notice that an explicit indirection
        LD      RØ,NN1;                         %  flag (I) is not needed by Z8000
        OUT     A_DATA,RØ                       % Send 16 bit word
        END;

MACRO   HOLD_REG PPØ,PP1;                       % Set Hold Reg of PPØ to PP1
        BEGIN
        POINT   PPØ,2;
        LD      RØ,PP1;
        OUT     A_DATA,RØ                       % Send 16 bit word
        END;


A>
```

The following definitions are provided for greater clarity in the C examples. These definitions are also utilized by the Am9513 evaluation program.

```
#define THEN                    /* ignore       */
#define BEGIN           {
#define END             }
#define RECORD          struct

#define CONTROL         0XDA    /* 9513 port addresses          */
#define DATA            0XD8


/*      master and counter fields       */

                        /* .FOUT_source .source values          */

#define TC_NM1          0       /* .source only                 */

#define SRC_1           1       /* Hardware source pins         */
#define SRC_2           2
#define SRC_3           3
#define SRC_4           4
#define SRC_5           5

#define GATE_1          6       /* Hardware gate pins           */
#define GATE_2          7
#define GATE_3          8
#define GATE_4          9
#define GATE_5          0XA

#define F1              0XB     /* Frequency scaler taps        */
#define F2              0XC
#define F3              0XD
#define F4              0XE
#define F5              0XF

#define BCD             1       /* .scaler modes                */
#define BINARY          0


/*      master fields           */

#define TOD_OFF         0       /* .day_mode values             */
#define TOD_50HZ        1
#define TOD_60HZ        2
#define TOD_100HZ       3

#define DISABLE         0       /* .compar1,2 values            */
#define ENABLE          1

#define ON              0       /* .FOUT_gate .data_ptr values  */
#define OFF             1

#define BUS_8           0       /* .bus_width values            */
#define BUS_16          1
```

```
/*      counter fields                  */

#define OFF_LO_TC       0       /* .output values                   */
#define ACT_HI_TC       1
#define TC_TOGGLE       2
#define OFF_OC_TC       4
#define ACT_IO_TC       5

#define UP              1       /* .direction values                */
#define DOWN            0

/*      counter mode control modes      */

#define MODE_ABC        0       /* Modes A,B,C                      */
#define MODE_DEF        1       /* Modes D,E,F                      */
#define MODE_GHI        2       /* Modes G,H,I                      */
#define MODE_JKL        3       /* Modes J,K,L                      */
#define MODE_mNO        4       /* Modes N,O - M is illegal         */
#define MODE_pQR        5       /* Modes Q,R                        */
#define MODE_Stu        6       /* Mode S                           */
#define MODE_Vwx        7       /* Mode V       */

#define RISE            0       /* .edge values                     */
#define FALL            1

#define NO_GATE         0       /* .gate values                     */
#define HL_TC_NM1       1
#define HL_NP1_GATE     2
#define HL_NM1_GATE     3
#define HL_GATE_N       4
#define LL_GATE_N       5
#define HE_GATE_N       6
#define LE_GATE_N       7


/*      load data pointer fields        */

#define MODE_           0       /* .element values                  */
#define LOAD_           1
#define HOLD_           2
#define HOLD_CYCLE      3

#define ALARM1          0       /* .element values (CTRL_GROUP)             */
#define ALARM2          1
#define MASTER          2
#define STATUS          3

#define CTRL_GROUP      7       /* .group values (inc. counters 1-5)        */

#define LOCAL           register
#define ASCICR          0XD
#define ASCILF          0XA
#define UPPER_CASE      0XDF    /* mask for upper case              */
#define RD              0       /* read or write                    */
#define WR              1
#define SEQUENCE        5       /* types of access                  */
#define NOT_SEQUENCE    6
#define LEVEL           7       /* types of gating                  */
#define EDGE            8
```

The following definitions are provided for greater clarity in the assembler examples.

```
;  Am9513 equates file for macros - Am8080/Am8085/Z80 format

        ; Master and counter Mode register functions

                                        ; FOUT source and count source definitions
TC_NM1      EQU 0       ; TC N-1 is count source only
SRC_1       EQU 1       ; Hardware source pins
SRC_2       EQU 2
SRC_3       EQU 3
SRC_4       EQU 4
SRC_5       EQU 5

GATE_1      EQU 6       ; Hardware gate pins
GATE_2      EQU 7
GATE_3      EQU 8
GATE_4      EQU 9
GATE_5      EQU 0AH

F1          EQU 0BH     ; Frequency scaler tap points
F2          EQU 0CH
F3          EQU 0DH
F4          EQU 0EH
F5          EQU 0FH

BCD         EQU 1       ; Counting modes
BINARY      EQU 0


        ; Master register functions

TOD_OFF     EQU 0       ; Time of day counting modes
TOD_50HZ    EQU 1
TOD_60HZ    EQU 2
TOD_100HZ   EQU 3

DISABLE     EQU 0       ; Comparators 1 and 2 modes
ENABLE      EQU 1

ON          EQU 0       ; FOUT gate and data pointer values
OF          EQU 1

BUS_8       EQU 0       ; 8 or 16 bit bus
BUS_16      EQU 1
```

```
        ; Counter Mode register functions

                                        ; TC output modes
OFF_LO_TC    EQU 0      ; Output inactive, low
ACT_HI_TC    EQU 1      ; Active hi terminal count pulse
TC_TOGGLE    EQU 2      ; TC toggled
OFF_OC_TC    EQU 4      ; Inactive, high impedance
ACT_LO_TC    EQU 5      ; Active lo terminal count pulse

UP           EQU 1      ; Count directions
DOWN         EQU 0

MODE_ABC     EQU 0      ; Counter mode control codes - A,B,C
MODE_DEF     EQU 1      ; Modes D,E,F
MODE_GHI     EQU 2      ; Modes G,H,I
MODE_JKL     EQU 3      ; Modes J,K,L
MODE_mNO     EQU 4      ; Modes N,L - notice Mode M is illegal
MODE_pQR     EQU 5      ; Modes Q,R
MODE_Stu     EQU 6      ; Mode S
MODE_Vwx     EQU 7      ; Mode V

RISE         EQU 0      ; Active count edge values
FALL         EQU 1

NO_GATE      EQU 0      ; Counter gating modes
HL_TC_NM1    EQU 1      ; Active Hi Level TC N-1
HL_NP1_GATE  EQU 2      ; Active Hi Level Gate N+1
HL_NM1_GATE  EQU 3      ; Active Hi Level Gate N-1
HL_GATE_N    EQU 4      ; Active Hi Level Gate N
LL_GATE_N    EQU 5      ; Active Lo Level Gate N
HE_GATE_N    EQU 6      ; Active Hi Edge Gate N
IE_GATE_N    EQU 7      ; Active Lo Edge Gate N

MODE_        EQU 0      ; Data pointer element values
LOAD_        EQU 1
HOLD_        EQU 2
HOLD_CY      EQU 3

ALARM1_      EQU 0      ; Data pointer element values (CTRL_GR)
ALARM2_      EQU 1
MASTER_      EQU 2
STATUS_      EQU 3

CTRL_GR      EQU 7      ; Data pointer group values (inc. counters 1-5)
```

```
% Am9513 equates file for macros - Z8000 format

CONST

        % Master and counter Mode register functions

                                        % FOUT source and count source definitions
TC_NM1      =   0,      % TC N-1 is count source only
SRC_1       =   1,      % Hardware source pins
SRC_2       =   2,
SRC_3       =   3,
SRC_4       =   4,
SRC_5       =   5,

GATE_1      =   6,      % Hardware gate pins
GATE_2      =   7,
GATE_3      =   8,
GATE_4      =   9,
GATE_5      =   0AH,

F1          =   0BH,    % Frequency scaler tap points
F2          =   0CH,
F3          =   0DH,
F4          =   0EH,
F5          =   0FH,

BCD         =   1,      % Counting modes
BINARY      =   0,


        % Master register functions

TOD_OFF     =   0,      % Time of day counting modes
TOD_50HZ    =   1,
TOD_60HZ    =   2,
TOD_100HZ   =   3,

DISABLE     =   0,      % Comparators 1 and 2 modes
ENABLE      =   1,

ON          =   0,      % FOUT gate and data pointer values
OF          =   1,

BUS_8       =   0,      % 8 or 16 bit bus
BUS_16      =   1,
```

```
        % Counter Mode register functions

                                        % TC output modes
CFF_LC_TC     =   0,        % Output inactive, low
ACT_HI_TC     =   1,        % Active hi terminal count pulse
TC_TOGGLE     =   2,        % TC toggled
CFF_OC_TC     =   4,        % Inactive, high impedance
ACT_LO_TC     =   5,        % Active lo terminal count pulse

UP            =   1,        % Count directions
DOWN          =   0,

MODE_ABC      =   0,        % Counter mode control codes - A,B,C
MODE_DEF      =   1,        % Modes D,E,F
MODE_GHI      =   2,        % Modes G,H,I
MODE_JKL      =   3,        % Modes J,K,L
MODE_mNO      =   4,        % Modes N,L - notice Mode M is illegal
MODE_pQR      =   5,        % Modes Q,R
MODE_Stu      =   6,        % Mode S
MODE_Vwx      =   7,        % Mode V

RISE          =   0,        % Active count edge values
FALL          =   1,

NO_GATE       =   0,        % Counter gating modes
HL_TC_NM1     =   1,        % Active Hi Level TC N-1
HL_NP1_GATE   =   2,        % Active Hi Level Gate N+1
HL_NM1_GATE   =   3,        % Active Hi Level Gate N-1
HL_GATE_N     =   4,        % Active Hi Level Gate N
LL_GATE_N     =   5,        % Active Lo Level Gate N
HE_GATE_N     =   6,        % Active Hi Edge Gate N
IE_GATE_N     =   7,        % Active Lo Edge Gate N

MODE_         =   0,        % Data pointer element values
LOAD_         =   1,
HOLD_         =   2,
HOLD_CY       =   3,

ALARM1_       =   0,        % Data pointer element values (CTRL_GR)
ALARM2_       =   1,
MASTER_       =   2,
STATUS_       =   3,

CTRL_GR       =   7;        % Data pointer group values (inc. counters 1-5)
```